PERSONALIZED SKINCARE RECOMMENDATION SYSTEM

A MINI PROJECT REPORT

by

AABEL ABY (VJC22CS001)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISWAJYOTHI COLLEGE OF ENGINEERING AND TECHNOLOGY, VAZHAKULAM

APRIL 2025

PERSONALIZED SKINCARE RECOMMENDATION SYSTEM

A MINI PROJECT REPORT

by

AABEL ABY (VJC22CS001)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

under the guidance

of

Ms. SIMNA UMMER V U

Assistant Professor, CSE Department

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISWAJYOTHI COLLEGE OF ENGINEERING AND TECHNOLOGY, VAZHAKULAM

APRIL 2025

VISWAJYOTHI COLLEGE OF ENGINEERING AND TECHNOLOGY, VAZHAKULAM

Department of Computer Science and Engineering

Vision

Moulding socially responsible and professionally competent Computer Engineers to adapt to the dynamic technological landscape

Mission

1. Foster the principles and practices of computer science to empower life-long learning and build careers in software and hardware development.

2. Impart value education to elevate students to be successful, ethical and effective problem-solvers to serve the needs of the industry, government, society and the scientific community.

3. Promote industry interaction to pursue new technologies in Computer Science and provide excellent infrastructure to engage faculty and students in scholarly research activities.

Program Educational Objectives

Our Graduates

1. Shall have creative aid critical reasoning skills to solve technical problems ethically and responsibly to serve the society.

2. Shall have competency to collaborate as a team member and team leader to address social, technical and engineering challenges.

3. Shall have ability to contribute to the development of the next generation of information technology either through innovative research or through practice in a corporate setting

4. Shall have potential to build start-up companies with the foundations, knowledge and experience they acquired from undergraduate education

Program Outcomes

1. **Engineering knowledge**:Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis**:Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences

3. **Design / development of solutions**:Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety and the cultural, societal and environmental considerations.

4. **Conduct investigations of complex problems**:Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.

5. **Modern tool usage**:Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:**Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:**Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of and need for sustainable development.

8. **Ethics:**Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice

9. **Individual and team work:**Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary settings

10. **Communication:**Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and unread in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning**:Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes

1. Ability to integrate theory and practice to construct software systems of varying complexity
2. Able to Apply Computer Science skills, tools and mathematical techniques to analyse, design and model complex systems
3. Ability to design and manage small-scale projects to develop a career in a related industry.

VISWAJYOTHI COLLEGE OF ENGINEERING AND TECHNOLOGY, VAZHAKULAM

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that the mini project work entitled **"PERSONALIZED SKINCARE RECOMMENDATION SYSTEM"** is a bonafide work done by **AABEL ABY (VJC22CS001)** in partial fulfillment of the award of the Degree of **Bachelor of Technology in Computer Science & Engineering** from APJ Abdul Kalam Technological University, Thiruvananthapuram, Kerala during the academic year 2024-2025.

Internal Supervisor **External Supervisor**

Mini Project Coordinator **Head of Department**

ACKNOWLEDGEMENT

First and foremost , I thank **God Almighty** for His divine grace and blessings in making all these possible . May He continue to lead me in the years . It is my privilege to render my heartfelt thanks to our most beloved Manager **Msg. Dr. PIOUS MALEKANDATHIL** , our Director **Rev. Fr. PAUL PARATHAZHAM** and our Principal **Dr. K K RAJAN** for providing me the opportunity to do this mini project during the sixth semester of my B.Tech degree course . I am deeply thankful to our Head of the Department , **Dr. AMEL AUSTINE** for his support and encouragement . I would like to express my sincere gratitude and heartfelt thanks to my Mini Project Guide **Ms. SIMNA UMMER V U** , Asst. Professor , Department of Computer Science and Engineering for her motivation , assistance and help for the project . I also express my sincere thanks to the Mini Project Coordinator **Ms. Neha Beegam P E** Asst. Professor, Department of Computer Science and Engineering for her guidance and support. I also thank all the staff members of the Computer Science Department for providing their assistance and support. Last, but not the least, I thank all my friends and family for their valuable feedback from time to time as well as their help and encouragement.

DECLARATION

I undersigned hereby declare that the mini project report **"PERSONALIZED SKINCARE RECOMMENDATIONS"**, submitted for partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology of the APJ Abdul Kalam Technological University is a bonafide work done under the supervision of **Ms. SIMNA UMMER V U**. This submission represents ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Vazhakulam
Date: 04-04-2025

AABEL ABY

ABSTRACT

This paper presents a comprehensive skincare recommendation platform that provides personalized product suggestions based on user-specific skin types (dry, oily, combination) and skin concerns (acne, hyperpigmentation, etc.). The system features a responsive HTML/CSS frontend interface that collects user inputs and displays results, connected to a robust Django-Python backend with SQLite database for efficient data management. Product recommendations are generated through structured database queries that match user profiles with pre-categorized skincare products, enabling precise filtering by skin compatibility and specific concerns. The platform includes administrative dashboards for managing product databases, skin type classifications, and concern categories, ensuring up-to-date and accurate recommendations. By implementing this systematic approach to skincare selection, the system eliminates guesswork and provides users with targeted product suggestions tailored to their individual needs..

Key Words :– Skincare Recommendations,Ingredient Checking,Skincare News, Django Framework, SQLite Database, Profile Matching.

Contents

Acknowledgement i

Declaration ii

Abstract iii

List of figures vii

1 INTRODUCTION 1

1.1 Problem Definition 1

1.2 Objective 2

1.3 Scope 3

2 EXISTING SYSTEM 5

2.1 Content-Based Filtering System 5

2.1.1 Advantages 5

2.1.2 Disadvantages 6

2.2 Deep Learning-Based Skincare Product Recommendation System 6

2.2.1 Advantages 6

2.2.2 Disadvantages 7

2.3 Intelligent Facial Skincare Recommendation WebApp 7

2.3.1 Advantages 7

2.3.2 Disadvantages 8

3 REQUIREMENT ANALYSIS **9**

3.1 Product Perspective 9

3.2 Product Functions 9

3.2.1 User Application Functions: 9

3.2.2 Admin Application Functions: 10

3.3 User Classes and Characteristics 10

3.4 Hardware and Software Requirements 11

3.4.1 Hardware Requirements 11

3.4.2 Software Requirements 11

3.4.3 Visual Studio Code 11

4 SYSTEM DESIGN **13**

4.1 Context Viewpoint Design 13

4.2 Information Viewpoint Design 16

4.3 Interface Viewpoint Design 17

4.4 Interaction Viewpoint Design 19

5 PROJECT WORK PLANNING **25**

5.1 Gantt Chart 25

6 IMPLEMENTATION AND TESTING **26**

6.1 Implementation 26

6.1.1 Algorithms Used 27

6.1.2 Database Structures 28

6.2 Testing 33

6.2.1 Test Cases for Algorithm to Generate Personalized Skincare 33

6.2.2 Test Cases for Algorithm to Manage Products from Admin Side: 33

6.2.3 Test Cases for Algorithm to View Recommendations on User Side: 33

7 CONCLUSION AND FUTURE SCOPE **34**

7.1 Conclusion 34

7.2 Future Scope 34

References **35**

Appendix A Implementation code **37**

A.1 Admin Side 37

A.2 User Side 48

Appendix B Screenshots **54**

B.1 Admin Home Page 54

B.2 Admin Product Entry 54

B.3 Admin Skin Type Entry 55

B.4 Admin Skin concern Entry 55

B.5 Admin Complaint Page 56

B.6 User Home Page 56

B.7 User Sign Up Page 57

B.8 User Login 58

B.9 Skincare Product Recommendations 59

B.10Ingredient Checker 59

B.11User Complaint 60

B.12Skincare News 60

B.13Skincare Tips 61

List of Figures

3.1 Use Case Diagram 10

4.1 Visualization of Sign Up Use Case Diagram for Personalized Skincare Recommendations 13

4.2 Visualization of Login Use Case Diagram for Personalized Skincare Recommendations 14

4.3 Visualization of Select Skintype Use Case Diagram for Personalized Skincare Recommendations 14

4.4 Visualization of Specify Skin Problem Use Case Diagram 14

4.5 Visualization of Personalized Skincare Recommendations Use Case Diagram 14

4.6 Visualization of View Product Details Use Case Diagram 15

4.7 Visualization of Routine Builder Use Case Diagram 15

4.8 Visualization of Skincare News Use Case Diagram 15

4.9 Visualization of Skincare Tips Use Case Diagram 15

4.10 Visualization of Manage Users Use Case Diagram 16

4.11 Visualization of Manage Products Use Case Diagram 16

4.12 Visualization of ER Diagram 16

4.13 Visualization of f System Component Diagram 18

4.14 Visualization of f User Component Diagram 18

4.15 Visualization of Server Component Diagram 19

4.16 Visualization of f Sign Up of User Sequence Diagram 19

4.17 Visualization of Login of User Sequence Diagram 20

4.18 Visualization Select SkinType Sequence Diagram 20

4.19 Visualization of Specify Skin Problem sequence Diagram 21

4.20 Visualization of Get Recommendation Sequence Diagram 21

4.21 Visualization of View Product Details Sequence diagram 21
4.22 Visualization of Routine Builder Sequence Diagram 22
4.23 Visualization of Skincare News Sequence Diagram 22
4.24 Visualization of Skincare Tips Sequence Diagram 23
4.25 Visualization of Manage User Sequence Diagram 23
4.26 Visualization of Manage Product Sequence Diagram 24
6.1 Admin 29
6.2 Product 29
6.3 Skin Type 30
6.4 Skin Concern 30
6.5 News 31
6.6 Question And Answer 32
B.1 Admin Dashboard 54
B.2 Admin-Product Entry 54
B.3 Admin-Skin Type Entry 55
B.4 Admin-Skin Concern Entry 55
B.5 Complaint 56
B.6 User Home Page 57
B.7 Sign Up Page 58
B.8 Login Page 58
B.9 Skincare Product Recommendations 59
B.10Check Ingredient 59
B.11User Complaints 60
B.12Skincare news 60
B.13Skincare Tips 61

Chapter 1

INTRODUCTION

In today's world, skincare has become a crucial part of our daily routines, but with so many products available, it's easy to feel overwhelmed. Choosing the right skincare products for your skin type and specific concerns can be confusing, leading to trial and error, wasted time, and sometimes even irritation. The problem is even more challenging when you don't know exactly what your skin needs or when you're unsure which product will work best for your unique skin.

This project aims to solve that problem with the Personalized Skincare Recommendation System—a simple, user-friendly way to get tailored skincare advice. The idea is to help people make better, more informed choices about their skincare routine based on their specific skin type (dry, oily, combination) and any particular concerns they might have, like acne, pigmentation, stretch marks, or warts.

Once users provide some basic information about their skin and concerns, the system will analyze the data and suggest skincare products that are best suited for them. The recommendations include things like face washes, moisturizers, sunscreens, and even treatments for specific issues. It'll also provide easy-to-understand instructions on how to use each product and links to purchase them.

The goal is to make skincare easier for everyone—helping users find the right products, save time, and feel more confident in their skin. By offering a personalized, hassle-free experience, the Personalized Skincare Recommendation System will make it easier for people to improve their skin health without the guesswork.

1.1 Problem Definition

The skincare industry is flooded with a vast range of products, making it difficult for consumers to identify the most suitable options for their unique skin types and concerns. Many individuals rely on trial-and-error methods, which often result in unnecessary expenses, ineffective treatments, and potential adverse effects on skin health. Additionally, existing skincare recommendation systems

typically provide generalized advice rather than personalized solutions tailored to an individual's specific needs, including skin type, age, gender, lifestyle, climatic conditions, and skin concerns such as acne, hyperpigmentation, sensitivity, and premature aging.

With the growing awareness of skincare and the increasing demand for customized beauty solutions, consumers expect more accurate, efficient, and trustworthy guidance when choosing products. However, the lack of centralized, user-friendly, and scientifically-driven platforms often leads to confusion, dissatisfaction, and low user engagement. Most available systems do not consider evolving skin conditions or integrate feedback mechanisms to improve recommendations over time.

Moreover, e-commerce platforms and dermatology services lack an intelligent and accessible mechanism to assist users in selecting the right skincare products efficiently. These platforms often fail to offer contextual and dynamic suggestions based on changing skin conditions, seasonal variations, or newly introduced products. There is also a noticeable gap in educating users about product ingredients, side effects, compatibility with other products, and correct application sequences.

The personalized skincare recommendation system seeks to bridge this gap by leveraging a structured, data-driven approach to generate customized suggestions. By combining user inputs with an expert-curated product database and rule-based algorithms, the system can provide relevant recommendations tailored to each user's profile. It also includes features like skincare routine builders, educational resources, product filtering options, and secure profile management to enhance the overall experience.

Through an intuitive web-based platform, users can receive scientifically informed skincare solutions, build personalized routines, compare products based on multiple parameters, and make better-informed purchasing decisions. The system ultimately aims to simplify the skincare journey, reduce dependency on trial and error, and promote healthier skin outcomes, while also supporting integration opportunities with e-commerce platforms and dermatology practices.

1.2 Objective

The objective of the Personalized Skincare Recommendation System is to assist users in selecting the most appropriate skincare products based on their individual skin types, concerns, and preferences. The system aims to eliminate the common trial-and-error approach in skincare selection by offering tailored product recommendations through a user-friendly web-based platform.

The system is designed to provide a centralized and intelligent platform that collects and analyzes user input such as age, gender, skin type (e.g., dry, oily, combination), and specific concerns (e.g., acne, hyperpigmentation, sensitivity). Based on this data, the system generates personalized

skincare product suggestions that match the user's needs.

One of the key objectives is to streamline user experience and decision-making by offering categorized product options, detailed information about ingredients, usage, and suitability, and allowing users to build a customized skincare routine. This reduces confusion, enhances product satisfaction, and contributes to better skincare outcomes.

The system also aims to support educational features, providing users with skincare tips, articles, and news that help them make informed decisions and improve their skincare knowledge.

Furthermore, the software promotes data-driven analysis and product filtering by enabling users to filter recommendations by brand, price range, or product type. This adds flexibility and personalization to the user experience.

Another essential goal is to ensure data privacy and security. The system is designed with robust authentication and secure data handling mechanisms to protect sensitive user information.

Overall, the Personalized Skincare Recommendation System strives to create a reliable, personalized, and interactive skincare support platform that empowers users to make the right choices for their skin health, enhances product usability, and integrates smoothly with potential e-commerce solutions in the future.

1.3 Scope

The scope of the Personalized Skincare Recommendation System encompasses the design, development, and deployment of a web-based application aimed at providing users with personalized skincare product recommendations based on individual skin characteristics and concerns. The system is intended to serve as an intelligent digital assistant that simplifies the skincare selection process by replacing the traditional trial-and-error approach with a data-driven, user-centric solution.

The system will collect user inputs such as skin type (e.g., oily, dry, combination), and specific skincare concerns (e.g., acne, hyperpigmentation, sensitivity), and use predefined rules and a structured database to generate personalized recommendations. It will also allow users to build customized skincare routines and receive guidance on product usage and suitability.

The software includes core modules such as user registration and login, skin type selection, skincare concern identification, recommendation engine, routine builder, and educational resources. It provides additional features such as product details, reviews, news updates, skincare tips, and secure profile management. The application is designed with a user-friendly interface accessible across multiple devices including desktops, tablets, and mobile phones.

Beyond the initial implementation, the system offers scalability for future enhancements such as integration with e-commerce platforms, dermatology clinics, and APIs for real-time product

updates. It also opens opportunities for incorporating advanced features like skin image analysis, AI-based recommendation models, user feedback mechanisms, and community-driven content sharing.

The system is applicable to a wide range of users, including general consumers seeking skincare guidance, skincare enthusiasts looking for structured routines, and businesses aiming to offer personalized product suggestions. It also holds potential for use in dermatological advisory settings, offering a decision-support tool for professionals.

Overall, the scope of the Personalized Skincare Recommendation System extends beyond basic product matching, aiming to provide a comprehensive, personalized, and interactive skincare experience that improves skin health outcomes and enhances user satisfaction through informed decision-making.

Chapter 2

EXISTING SYSTEM

Several skincare recommendation systems have been developed to address the growing demand for personalized solutions. This section examines three notable systems from recent literature, detailing their methodologies, strengths, and limitations, and comparing them to our input-based approach.

2.1 Content-Based Filtering System

This system utilizes content-based filtering to recommend skincare products by analyzing ingredient similarity and user skin types. It collects data from Sephora (1472 products), including brand, price, ingredients and skin type compatibility (e.g., dry, oily), scraped, and processed using Natural Language Processing (NLP). Users select an initial product, and the system applies cosine similarity to suggest the top five matches from six product categories (e.g., moisturizers, cleansers). Although it emphasizes ingredient-driven personalization, its dependency on an initial user choice and lack of broader skin concern analysis limit its scope compared to our system, which relies on user inputs like skin type and concerns without requiring prior product knowledge.

2.1.1 Advantages

- Ingredient-focused precision: By focusing on ingredients, this system ensures that recommendations meet specific skin needs, such as choosing a dish based on what your body can handle. For example, if you have oily skin, it might suggest something oil-controlling based on chemical makeup.
- Structured and data-driven: With 1472 products from Sephora, it is built on a solid dataset, making it reliable for users who know their skin type. It's like having a well-stocked library to pull recommendations from.
- Cross-category potential: It covers six product types, so it's not stuck recommending just one

thing (e.g., only moisturizers). This gives users a broader skincare routine—like a full meal plan instead of just a snack.

2.1.2 Disadvantages

- Requires an initial product choice: This system needs you to pick a starting product, which can be a hurdle if you're new to skincare or unsure what works—like being asked to name a favorite book before you've read anything.
- No validation measure: This system has no accuracy check (e.g., purity or satisfaction rate), so you're trusting it works without proof—like buying a gadget with no reviews.
- Overspecialization risk: Since it matches based on ingredient similarity, it might keep suggesting similar stuff (e.g., all cleansers) and miss complementary products.

2.2 Deep Learning-Based Skincare Product Recommendation System

This system uses deep learning to recommend skincare products by analyzing cosmetic ingredients and user skin conditions. A transformer encoder processes ingredient lists (treated as sequential data) to predict the efficacy of 18 cosmetic effects (e.g. anti-aging, moisturizing). A U-Net-based skin analyzer evaluates six skin concerns (pores, redness, acne, wrinkles, pigmentation, dark circles) from frontal face photos through LUMINI Kiosk V2. The recommendation module matches skin scores with product efficacy using cosine similarity, filtering by 12 skin types from surveys. It's tested on 8000 products and datasets like FFHQ and CelebA-HQ.

2.2.1 Advantages

- Smart ingredient analysis: The Transformer model digs into ingredient lists to predict effects like brightening or wrinkle reduction—like a skincare detective figuring out what each product can really do. For example, it nailed Product A's pigmentation benefits at 99.8% confidence.
- Tailored recommendations: By combining skin scores and ingredient efficacy, it picks products that match your exact needs across five categories (cleansers, serums, etc.)—think of it as a personal shopper who knows your skin inside out.
- Precise skin condition targeting: The U-Net skin analyzer spots specific issues (e.g. acne, pores) from photos with strong accuracy (correlation ¿0.7 with dermatologists)—it's like having a mini skin expert on your phone.

2.2.2 Disadvantages

- Relies on facial images: This system needs a photo from a device like LUMINI Kiosk V2, which isn't practical if you don't have that tech handy—imagine needing a special camera just to get started, when your system works fine with simple inputs.
- Data imbalance issues: The ingredient analyzer struggles with less common effects (e.g., acne precision at 33.3%) because the dataset favors popular tags like anti-aging, leaving some skin concerns underserved.
- Marketing tag dependency: The system relies on marketing claims as ground truth, which might be skewed by hype rather than real results—like trusting a flashy ad over a lab test. It admits this could be less objective without clinical data.
- Complex setup for users: Needing a photo, survey, and specific hardware makes it less user-friendly than your click-and-go approach—it's like asking someone to fill out a form and take a selfie when they just want quick advice.

2.3 Intelligent Facial Skincare Recommendation WebApp

This system is a web-based application that uses a CNN to analyze facial images uploaded by users, classifying skin types (normal, dry, combination, oily, sensitive) and conditions (scaly, red spots, skin moles). The CNN model is trained on a dataset of labeled facial images, preprocessed with resizing, normalization, and augmentation (e.g., rotation, flipping). It features multiple convolutional, pooling, and dense layers, optimized with Adam and binary cross-entropy loss. The webapp integrates this model into a user-friendly interface with a backend for real-time processing, offering tailored product recommendations based on the analysis. It achieves a test accuracy of 98.81%, validated with metrics like precision, recall, and F1-score.

2.3.1 Advantages

- High accuracy in skin classification: With a test accuracy of 98.81%, the CNN nails skin type and condition detection—like a skincare pro who rarely gets it wrong. It's reliable for spotting oily skin or red spots from a photo.
- Broad skin type and condition coverage: It handles five skin types and three conditions (scaly, red spots, moles), offering more variety than some systems.
- User-friendly web interface: The app's design lets anyone upload a picture and get advice without tech know-how—imagine a skincare buddy you can chat with online, no appointment needed.

- Real-time processing: The backend processes images and delivers recommendations instantly, so you're not left waiting.

2.3.2 Disadvantages

- Requires facial images: The users need to snap and upload a photo, which could be difficult if you don't have a good camera or lighting—think of it as needing a selfie stick when you just want quick advice.
- Limited to predefined categories: The five skin types and three conditions don't cover everything (e.g., no acne or wrinkles explicitly), so it might miss specific concerns your project includes.
- Potential overfitting risk: While it uses dropout layers (0.5 rate), the huge gap between training (high accuracy) and real-world use could mean it's too tuned to the test data.
- Dataset dependency: Its performance hinges on the quality and diversity of its facial image dataset, which isn't detailed in size or source.

Chapter 3

REQUIREMENT ANALYSIS

3.1 Product Perspective

The Personalized Skincare Recommendation System is a web-based platform designed to offer tailored skincare product recommendations. The system provides personalized advice based on users' skin types and specific concerns like acne, hyperpigmentation, stretch marks, and warts. Users can easily navigate the app to select their skin type and input their skincare issues to receive product suggestions, ensuring that they find the best products suited for their unique needs.

3.2 Product Functions

The core functionality of the Personalized Skincare Recommendation System revolves around providing users with a seamless experience for logging in, specifying their skin concerns, and receiving tailored product recommendations. Here's a breakdown of each feature:

3.2.1 User Application Functions:

Input collected from users:

- User Login/Registration.
- Collect personal information such as Age, Gender, Address etc.
- Skin type selection.
- Skin concern selection.
- Complaints about products.

Output Provided to users:

- Personalized Skincare Recommendations.

- Skin care tips and routine.
- The latest skincare news.
- Reply from admin.

3.2.2 Admin Application Functions:

Input collected from Admin

- Login Credentials.
- Manage the products.
- Update news.
- Reply to the complaints.

Output Provided to Admin:

- Information on registered users.

3.3 User Classes and Characteristics

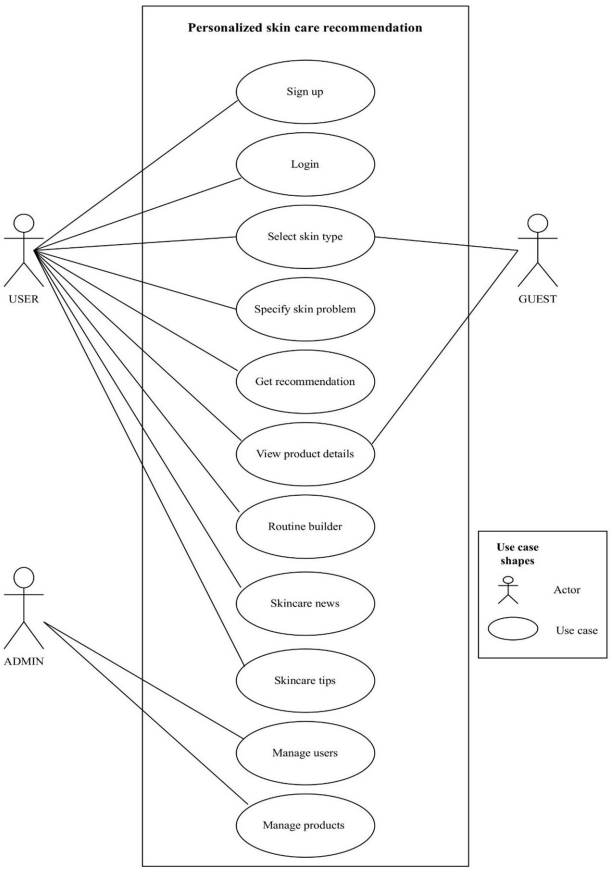

Figure 3.1: Use Case Diagram

- Admin
  - Stores user info securely in the database.
  - Generates custom skincare recommendations based on the user's inputs.
- User
  - Users can create an account with their name, email, and password.
  - They can select their skin type and skin concern.
  - They receive personalized product recommendations.

3.4 Hardware and Software Requirements

3.4.1 Hardware Requirements

For the hardware interfaces, the system necessitates an internet connection to facilitate the application's connectivity to the web. Additionally, users must have access to a device such as a desktop, laptop, tablet, or mobile phone running on either the Android or iOS operating system to utilize the application effectively.

3.4.2 Software Requirements

The software requirements for implementing the front-end and back-end of the Personalized Skincare Recommendation System include React.js for the front-end, while Node.js or Python (Django/Flask) will handle the back-end logic. Firebase Authentication will ensure secure user authentication through email and password, providing a reliable and safe login mechanism. The SQlite database will be used to store user data, skincare preferences, and product recommendations, ensuring seamless real-time synchronization and efficient retrieval. The system is optimized for use on the web, compatible with all modern browsers (such as Chrome, Firefox, and Safari) running on both Windows and MacOS. The system's development will primarily utilize JavaScript (for React.js) and Node.js/Python for the backend functionalities. Authentication will be handled using JWT or OAuth for secure user sessions.

3.4.3 Visual Studio Code

Visual Studio Code (VS Code) stands out as a versatile and user-friendly code editor, crafted by Microsoft to cater to the diverse needs of developers. With its sleek and intuitive interface, VS Code provides a seamless coding experience, empowering developers to focus on their craft without unnecessary distractions. Its extensive ecosystem of extensions amplifies its capabilities,

allowing users to tailor the editor to their specific workflows and preferences. Whether working on small scripts or large-scale projects, VS Code's robust features, including built-in Git integration, IntelliSense, debugging support, and task automation, streamline the development process and boost productivity.

Chapter 4

SYSTEM DESIGN

4.1 Context Viewpoint Design

Our personalized skincare recommendation system caters to three types of users: guests, regular users, and admins. Guests have limited access and can only sign up or log in to explore the system. Once logged in, regular users can select their skin type, specify concerns, and receive personalized skincare recommendations. They can also view detailed product information, create customized skincare routines, and stay informed with relevant skincare updates. Admins, on the other hand, have advanced privileges to manage user accounts, update product information, and maintain system content, ensuring that the recommendations provided remain accurate and helpful. This structure ensures that each user experiences a smooth and personalized journey based on their role.

More detailed information about use cases can be found in the system requirement specification document.

Sign Up

Users need to be signed up in order to benefit from the website. They need to specify their unique username and password. They are required to enter their corresponding personal information.

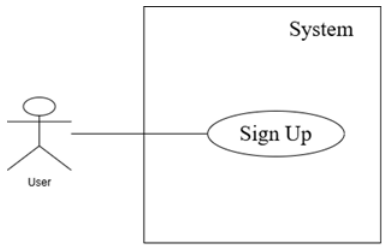

Figure 4.1: Visualization of Sign Up Use Case Diagram for Personalized Skincare Recommendations

Login

Users may log in to the system by entering their unique email ID and password on the main page.

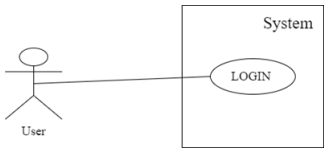

Figure 4.2: Visualization of Login Use Case Diagram for Personalized Skincare Recommendations

Select Skintype

The Select Skin Type page allows users to choose their skin type easily, ensuring personalized recommendations and a seamless experience.

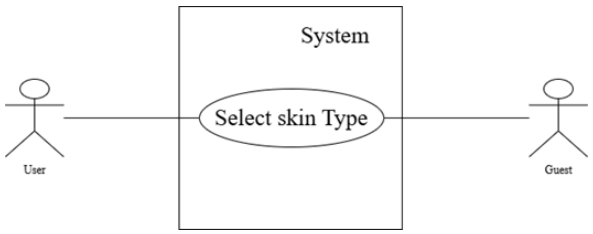

Figure 4.3: Visualization of Select Skintype Use Case Diagram for Personalized Skincare Recommendations

Specify Skin Problems

The Specify Skin Problem page allows users to enter their skin concerns, helping the system provide relevant suggestions and solutions.

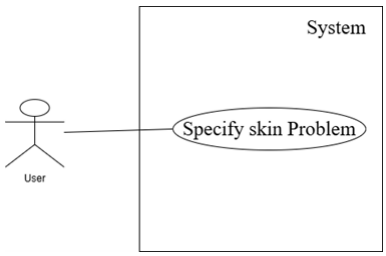

Figure 4.4: Visualization of Specify Skin Problem Use Case Diagram

Get Recommendations

The Get Recommendation feature provides users with personalized skincare suggestions based on their specified skin concerns

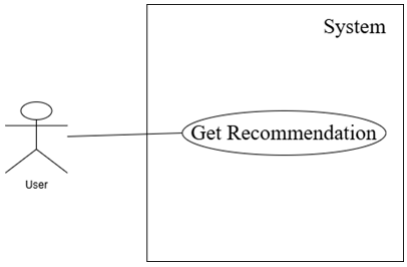

Figure 4.5: Visualization of Personalized Skincare Recommendations Use Case Diagram

View Product Details

The View Product Details feature allows both users and guests to access information about skincare products.

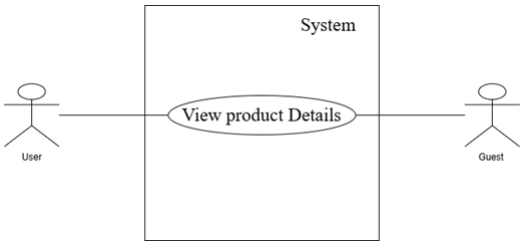

Figure 4.6: Visualization of View Product Details Use Case Diagram

Routine Builder

The Routine Builder feature allows users to create a personalized skincare routine based on their needs.

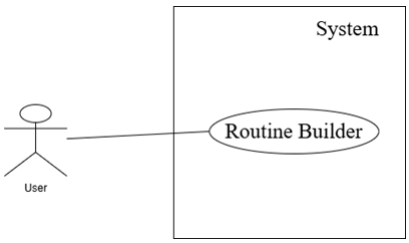

Figure 4.7: Visualization of Routine Builder Use Case Diagram

Skincare News

The Skincare News feature provides users with the latest updates, trends, and tips related to skincare.

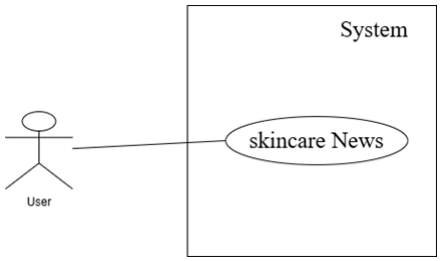

Figure 4.8: Visualization of Skincare News Use Case Diagram

Skincare Tips

The Skincare Tips feature provides users with helpful advice and best practices for maintaining healthy skin.

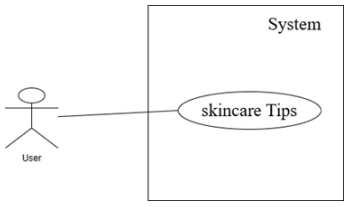

Figure 4.9: Visualization of Skincare Tips Use Case Diagram

Manage Users

The Manage Users feature allows the admin to oversee user accounts, update details, and manage access.

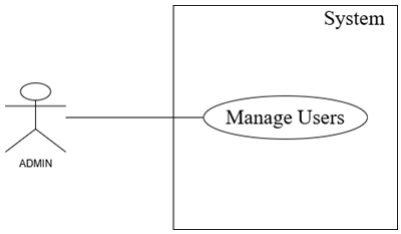

Figure 4.10: Visualization of Manage Users Use Case Diagram

Manage Products

The Manage Products feature enables the admin to add, update, or remove products from the system.

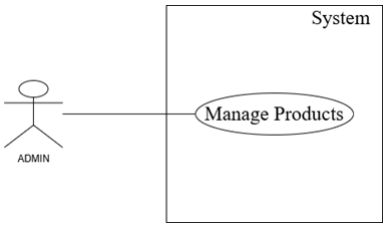

Figure 4.11: Visualization of Manage Products Use Case Diagram

4.2 Information Viewpoint Design

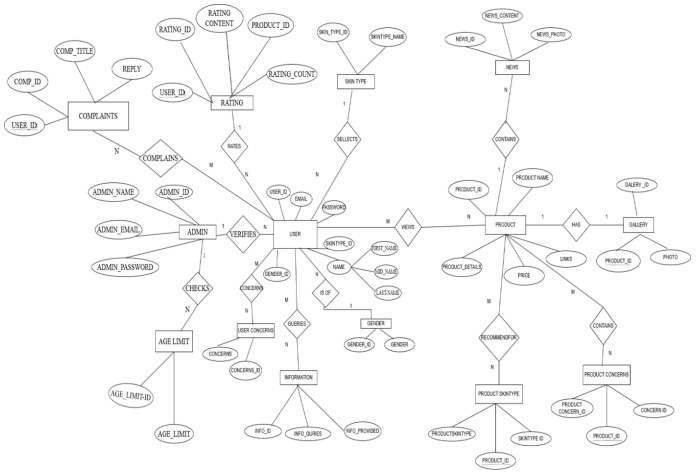

Figure 4.12: Visualization of ER Diagram

Design Entities

- User
- Admin
- Skin Type

- Product
- Product Concerns
- Complaints
- News
- Gallery
- Age Limit
- Information

Design Attributes

Table Name	Description
User	Holds the information of the user
Admin	Holds the information of the admin
Skin Type	Stores details about different skin types
Product	Holds product details and specifications
Product Concerns	Links products to specific skin concerns
Complaints	Holds user complaints and related responses
News	Contains skincare-related news and updates
Gallery	Stores related images and media
Age Limit	Holds age-related eligibility information
Information	Provides additional skincare tips and suggestions

Table 4.1: Design Attributes

4.3 Interface Viewpoint Design

System Interface

This viewpoint gives information about the correct usage of services provided by a design object. The system consists of two main components: User and server. The user interacts through the interface for selecting skin types and user management. The server processes these requests via the authentication service and recommendation engine, which retrieves relevant data from the database to provide personalized skincare recommendations.

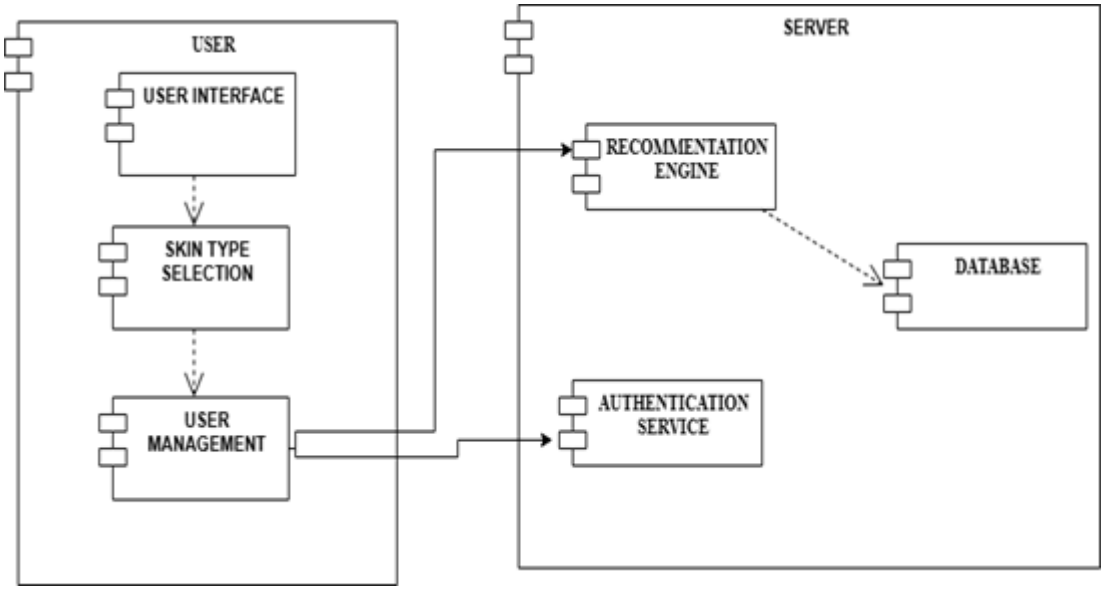

Figure 4.13: Visualization of f System Component Diagram

User Component

The User Component in our personalized skincare recommendation system aims to visualize events in the system and respond to user actions through the server. It consists of two main components: the User Interface, which facilitates user interactions, and User Management, which processes user data and preferences. User Management connects with the Server to handle events and ensure smooth system operations.

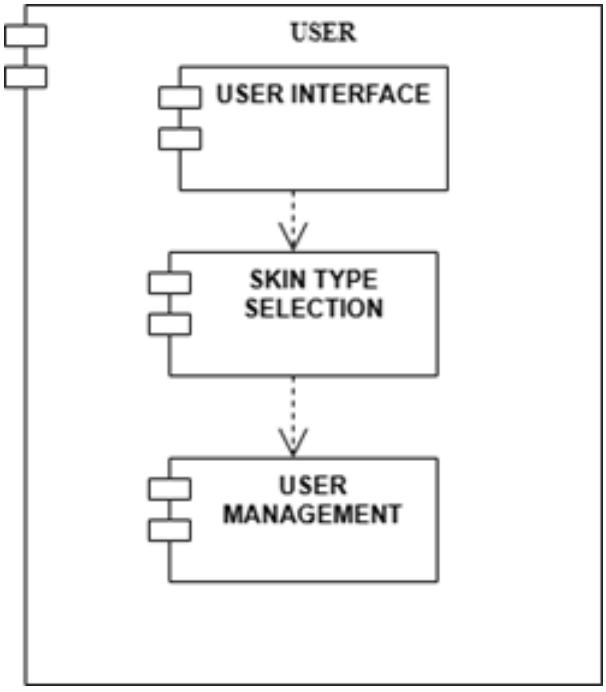

Figure 4.14: Visualization of f User Component Diagram

Server Component

The Server Component in our personalized skincare recommendation system manages data flow and controls server-side operations. It consists of two main components: the Recommendation Engine and the Authentication Service. The Recommendation Engine processes user input and retrieves relevant skincare data from the Database. The Authentication Service verifies user credentials and ensures secure access. The Database stores user information, product details, and skincare concerns to provide accurate recommendations.

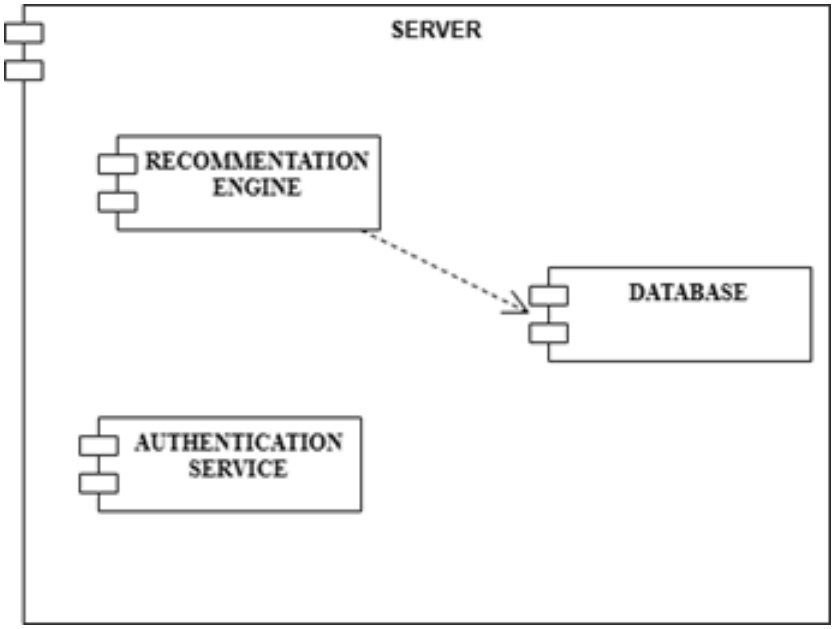

Figure 4.15: Visualization of Server Component Diagram

4.4 Interaction Viewpoint Design

Sign Up

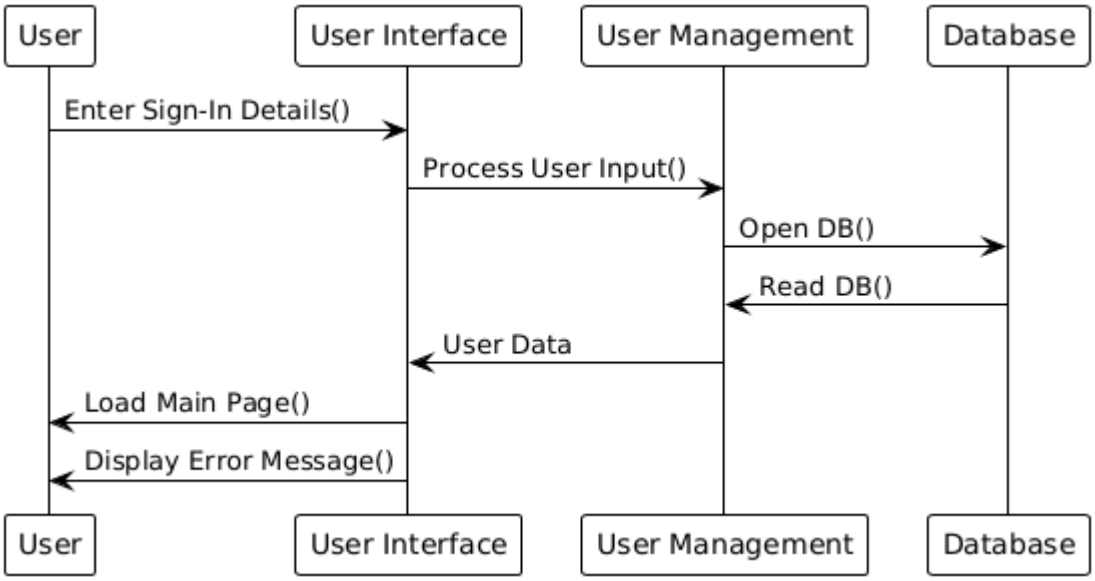

Figure 4.16: Visualization of f Sign Up of User Sequence Diagram

The user enters their sign-up details through the User Interface, which sends the information to User Management. User Management opens the Database to store the new user data. Once the data is saved successfully, the system either loads the main page or displays an error message if the process fails.

User Login

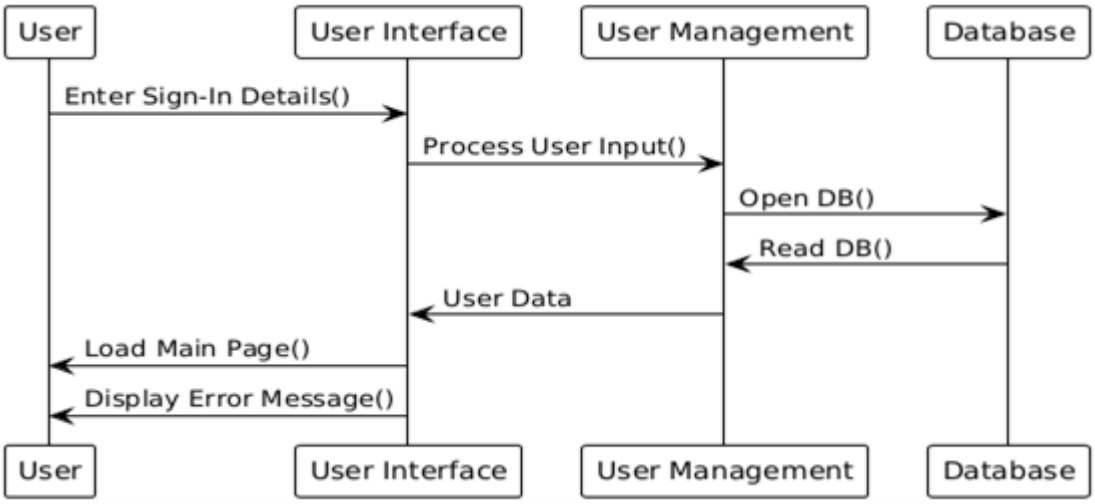

Figure 4.17: Visualization of Login of User Sequence Diagram

The user enters Login details through the interface, which are processed by the system. The system queries the database to verify the information. If valid, the main page loads; otherwise, an error message is displayed.

User Select SkinType

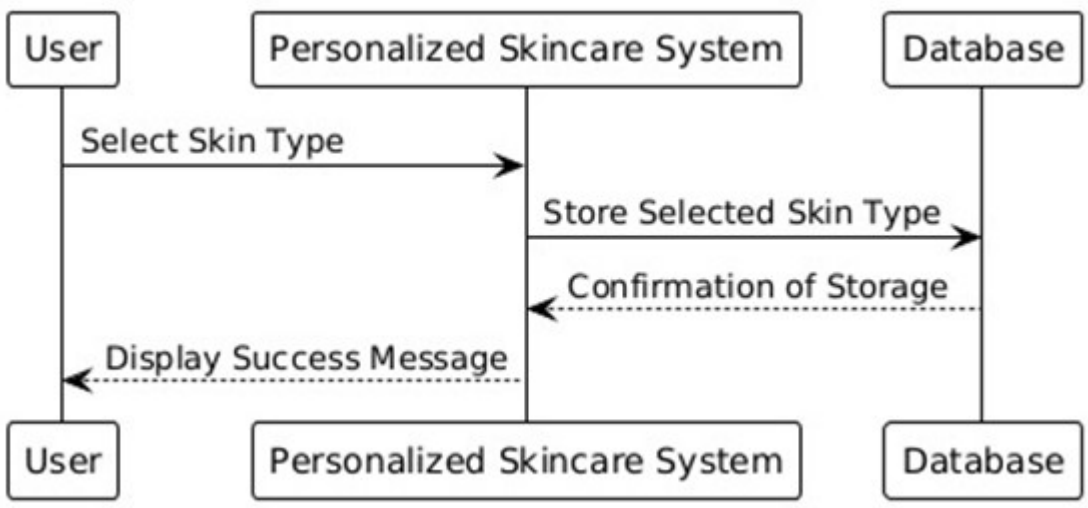

Figure 4.18: Visualization Select SkinType Sequence Diagram

The user selects their skin type, which is stored in the database by the system. Upon successful storage, a confirmation is sent, and a success message is displayed to the user.

Specify Skin Problem

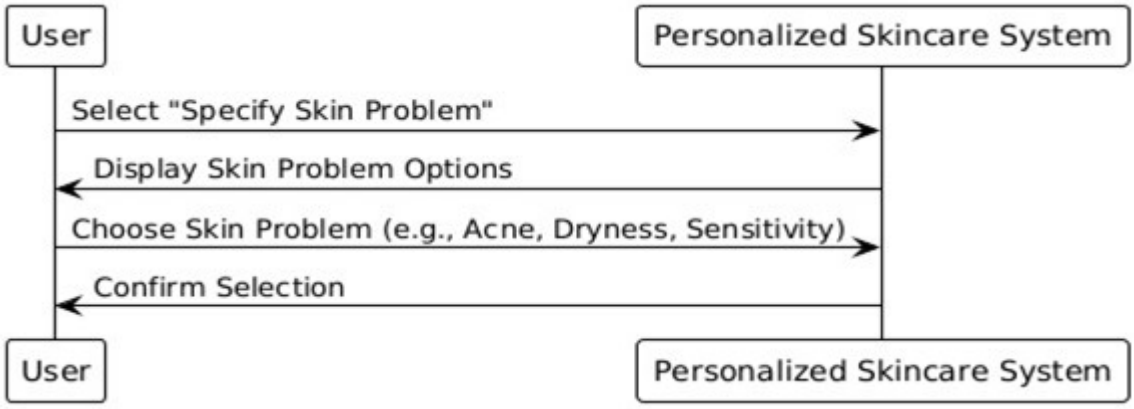

Figure 4.19: Visualization of Specify Skin Problem sequence Diagram

The user selects ”Specify Skin Problem,” and the system displays available options. The user chooses a specific problem (e.g., acne, dryness), and the system confirms the selection.

Get Recommendations

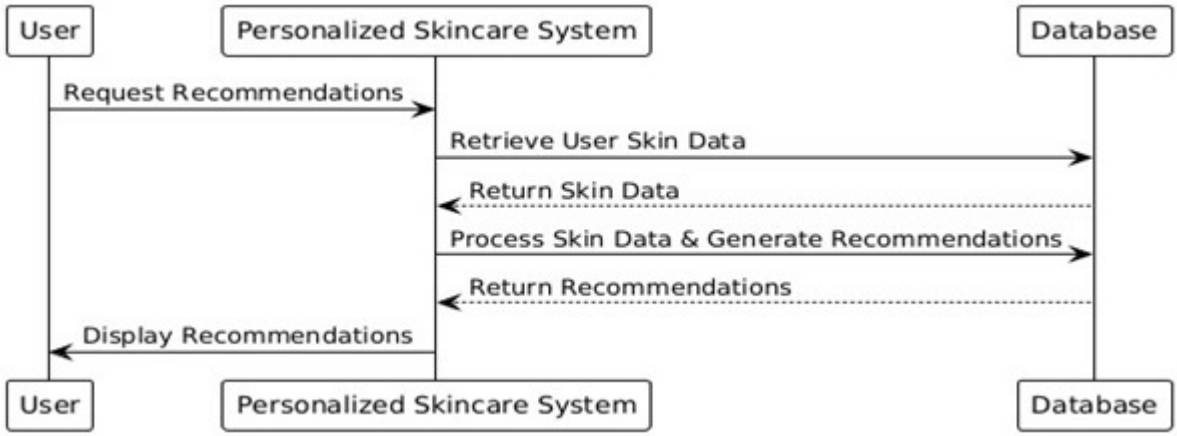

Figure 4.20: Visualization of Get Recommendation Sequence Diagram

The user requests recommendations, prompting the system to retrieve skin data from the database. The system processes the data, generates recommendations, and returns them to the user for display.

View Product Details

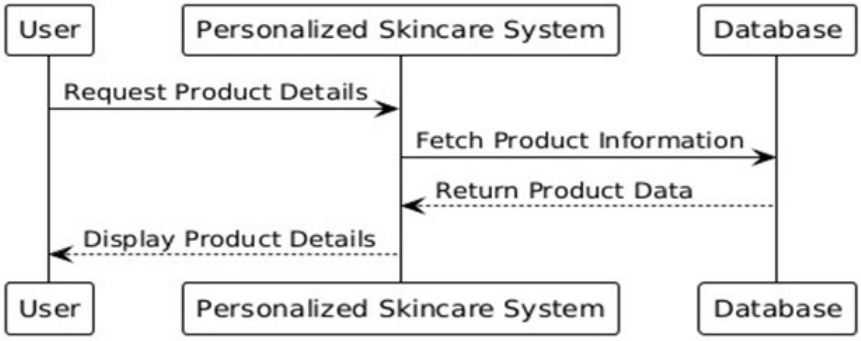

Figure 4.21: Visualization of View Product Details Sequence diagram

The user requests product details, prompting the system to fetch product information from the database. The system returns the product data and displays it to the user.

Routine Builder

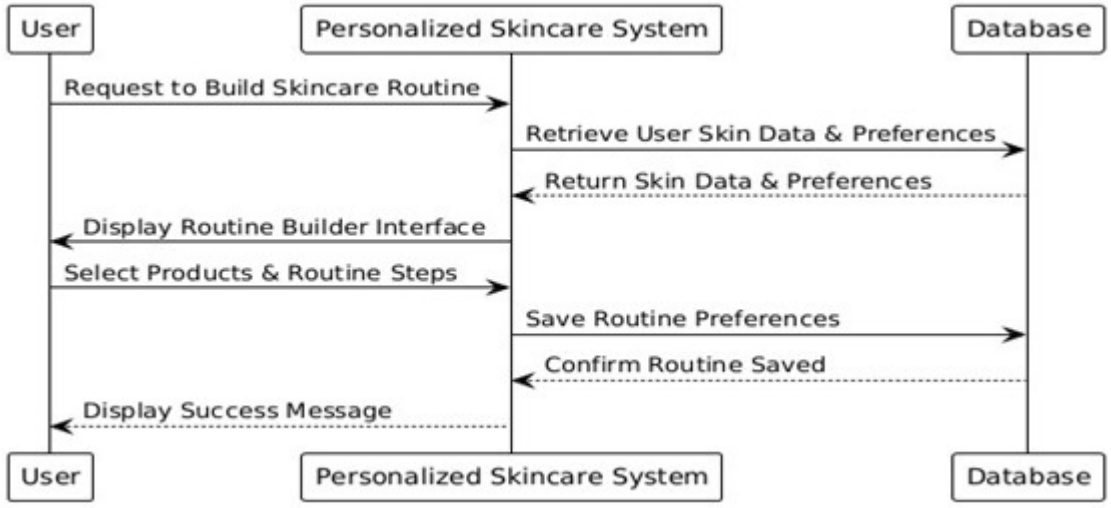

Figure 4.22: Visualization of Routine Builder Sequence Diagram

The user requests to build a skincare routine, prompting the system to retrieve skin data and preferences from the database. The system displays the routine builder interface where the user selects products and routine steps. The preferences are saved in the database, and a success message is displayed.

Skincare News

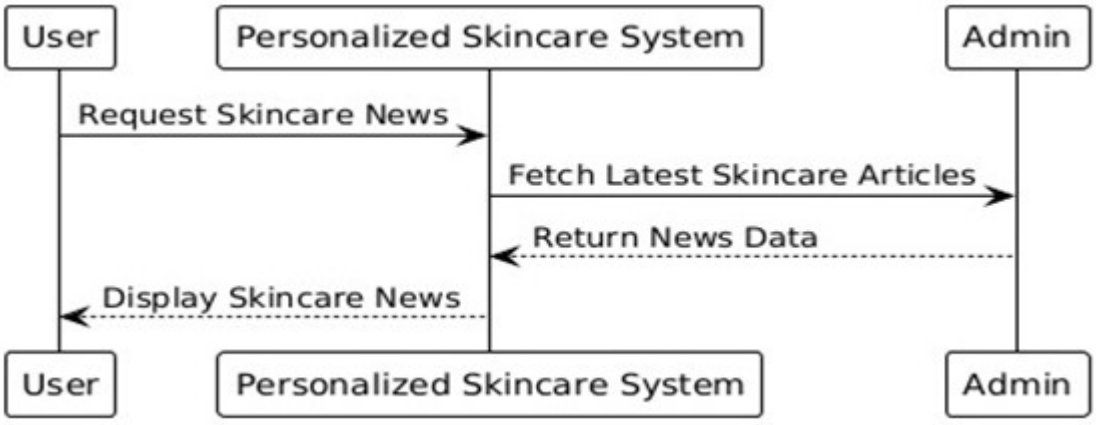

Figure 4.23: Visualization of Skincare News Sequence Diagram

The user requests skincare news, prompting the system to fetch the latest skincare articles from the admin. The admin returns the news data to the system, which then displays the skincare news to the user.

Skincare Tips

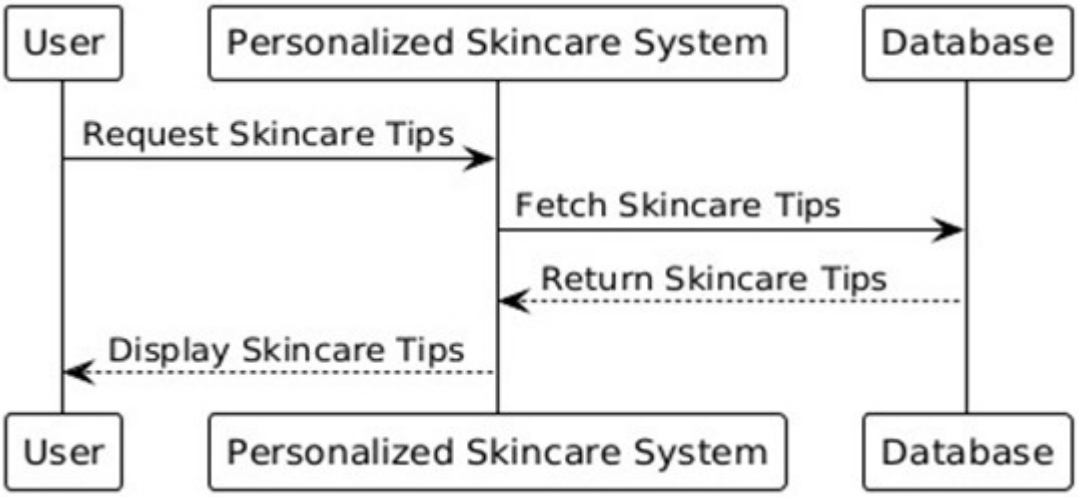

Figure 4.24: Visualization of Skincare Tips Sequence Diagram

The user requests skincare tips, prompting the Personalized Skincare System to fetch relevant tips from the database. The database returns the tips to the system, which then displays the skincare tips to the user.

Manage User

Figure 4.25: Visualization of Manage User Sequence Diagram

The admin sends a manage user request to the Personalized Skincare System, which updates the database. The database returns the status, and the system displays a confirmation message to the admin.

Manage Product

Figure 4.26: Visualization of Manage Product Sequence Diagram

The admin sends an add or delete product request to the Personalized Skincare System, which updates the database. The database returns the status, and the system displays a confirmation message to the admin.

Chapter 5

PROJECT WORK PLANNING

5.1 Gantt Chart

This Gantt chart outlines the timeline and progress of the ”Personalized Skincare Product Recommendations” project, which spans from December 17, 2024, to March 27, 2025. It breaks the project into key phases including literature review, abstract creation, requirement analysis, system and software design, development, testing, and final documentation. Each task is assigned a start and end date along with a progress percentage, visually represented by bars. The chart helps track overlapping tasks, dependencies, and ensures that the project progresses in a structured and timely manner.

GANTT CHART

Personalized Skincare Recommendations (PROJECT NAME) — 17/12/2024 (START DATE) — 27/03/2025 (END DATE)

Task ID	Task Name	Start Date	End Date	% Complete
1	**Review of Literature and Identification of problem, Team Creation**	17-12-2024	26-12-2024	100%
2	**Creation of Abstract**	27-12-2024	31-12-2024	100%
3	**Preparation of Software Requirement Specification**	11-01-2025	29-01-2025	100%
4	**Creation of Software Design Document**	07-02-2025	27-02-2025	100%
4.1	System Architecture Design	07-02-2025	08-02-2025	100%
4.2	Application of Architecture Design	09-02-2025	14-02-2025	100%
4.3	GUI Design	15-02-2025	17-02-2025	100%
4.4	API Design	18-02-2025	20-02-2025	100%
4.5	Database Design	21-02-2025	24-02-2025	100%
4.6	Technology Stack	27-02-2025	27-02-2025	100%
5	**Create Test Plan, Test Case, Traceability Matrix**	14-02-2025	17-02-2025	100%
6	**Creation of Project Plan**	17-02-2025	20-02-2025	100%
7	**Development**	21-02-2025	10-03-2025	100%
7.1	Set coding Standards	21-02-2025	22-02-2025	100%
7.2	Environment Setup	23-02-2025	23-02-2025	100%
7.3	Source Code Control Setup	24-02-2025	24-02-2025	100%
7.4	Development	25-02-2025	27-02-2025	100%
7.5	Unit Testing	27-02-2025	27-02-2025	100%
7.6	Integration Testing	28-02-2025	01-03-2025	100%
7.7	Testing/Quality Assurance	02-03-2025	03-03-2025	100%
7.8	Functional Testing	04-03-2025	05-03-2025	100%
7.9	Load Testing	06-03-2025	07-03-2025	100%
7.10	Report Bugs	08-03-2025	08-03-2025	100%
7.11	Resolve Bugs and Retest	09-03-2025	10-03-2025	100%
8	**Deployment**	11-03-2025	14-03-2025	100%
9	**Final Project Report**	13-03-2025	27-03-2025	100%

Chapter 6

IMPLEMENTATION AND TESTING

6.1 Implementation

The Personalized Skincare Recommendation System is implemented through a web-based application for users and admins, developed using Python with Django and SQLite as the database. The application connects users and admins to a shared SQLite database for efficient data management. During registration, users provide personal details such as name, email, age, gender, address, contact number, and password. On the admin side, admins register with details like name, email, and password to manage the system.

On the product recommendation page, users select their skin type (e.g., Dry, Oily, Combination) and concerns (e.g., Acne, pigmentation) from admin-provided lists. The system processes these inputs using predefined rules to generate personalized product recommendations (e.g., face wash, moisturizer). It assesses the suitability of recommendations by checking for potential risks based on ingredient compatibility. If a recommended product has no known issues for the user's skin type, a low-risk status is assigned, and no alert is generated. If the product contains ingredients that may cause minor irritation (e.g., high Vitamin C for Oily skin), a moderate-risk alert is triggered, advising a patch test. If the product includes ingredients likely to cause severe reactions (e.g., harsh exfoliants for Combination skin), a high-risk alert is generated, recommending the user avoid the product.

Alerts are stored in the SQLite database in a dedicated table 'Alerts' linked to the user, with fields: alert nature (low/moderate/high), alert message (detailing the risk), time of alert, and user's selected skin type/concerns. Admins can view these alerts along with user details (e.g., name, age) to update products if needed. The system ensures effective skincare management by providing tailored recommendations, identifying potential risks, and maintaining secure data storage in SQLite.

6.1.1 Algorithms Used

Algorithm to Get Product Recommendations:

1. Start
2. Authenticate the User
   - token ← GetJWTTokenFromRequest()
   - If Not VerifyJWTToken(token, user_id):
   - Redirect to ”/login”
   - Return
3. Query General Skincare Products (skindata)
   - skindata ← QueryDatabase(”SELECT id, product_name, product_details, product_price, product_amzonlink, product_flpctlink FROM products WHERE category = ’Skin Product’ AND skin_type = ?”,skin_type)
4. Query Concern-Specific Products (concerndata)
   - concerndata ← Empty List
   - If selected_concerns Is Not Empty:
   - all_concern_products ← QueryDatabase(”SELECT id, product_name, product_details, product_price, product_amzonlink, product_flpctlink FROM products WHERE category = ’Concern Product’)
   - For each product in all_concern_products:
   - For each concern in selected_concerns:
   - If concern In product.concerns:
   - Add product to concerndata
   - Break
5. Fetch Gallery Images for Products (for both skindata and concerndata)
   - For each product in skindata:
   - product.tbl_gallery_set ← QueryDatabase(”SELECT gallery_image FROM tbl_gallery WHERE product_id = ?”, product.id)
   - For each product in concerndata:
   - product.tbl_gallery_set ← QueryDatabase(”SELECT gallery_image FROM tbl_gallery WHERE product_id = ?”, product.id)

6. Render the Recommendations
    - RenderTemplate(”ProductSuggestion.html”, skindata=skindata, concerndata=concerndata)
7. Return the Results
    - Return skindata, concerndata
8. Stop

6.1.2 Database Structures

For the personalized skincare recommendation system, we have implemented SQLite as the main database due to its seamless integration with Django, lightweight architecture, and ease of setup. As a file-based relational database, SQLite eliminates the need for a separate server, making it an efficient choice for this project while maintaining structured data organization through tables, rows, and columns. Relationships between data are managed using foreign keys, ensuring referential integrity and logical connections across different datasets. The simplicity of SQLite and the single file storage make it an excellent fit for this application, particularly at this stage, where heavy scalability is not yet a requirement. Skin type, skin concern, product details, complaints, gallery, question and answer, news, and tips are stored as structured tables in the database. Each table contains organized records with clear relationships:

- Skin Type
- Skin Concern
- Product details
- Gallery
- Complaints
- News
- Tips
- Question And Answer

Database Screenshots

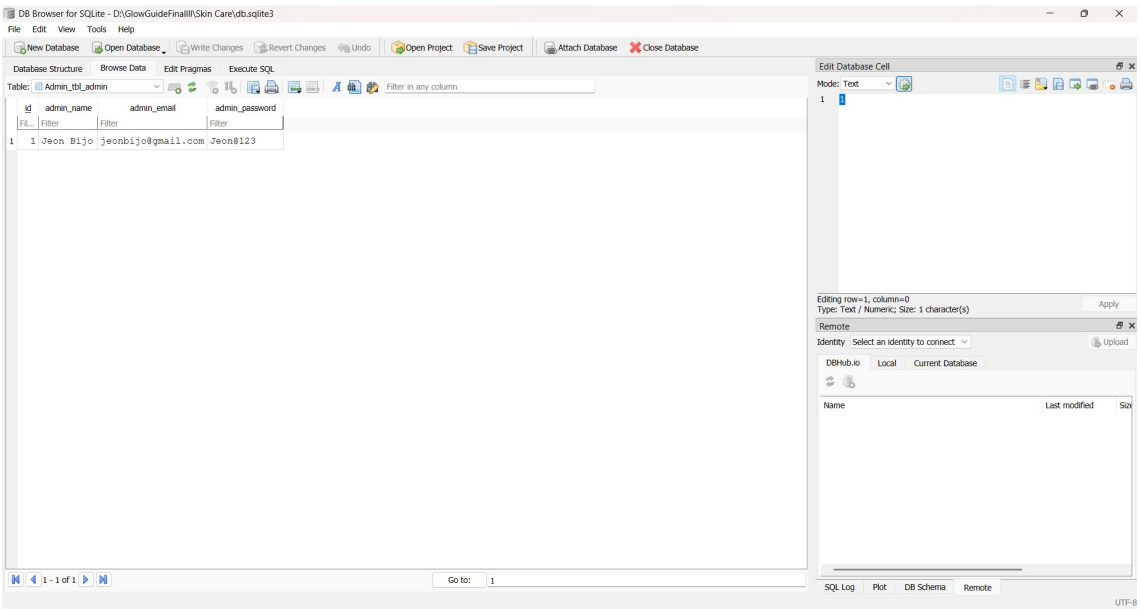

Figure 6.1: Admin

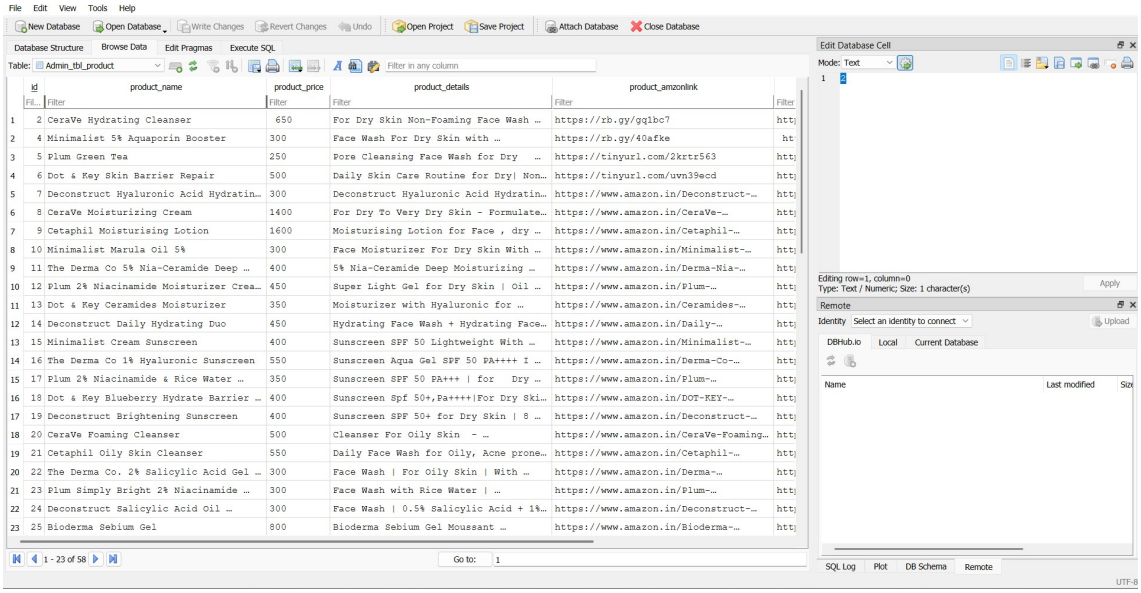
File Edit View Tools Help
New Database
Open Database
Write Changes
Revert Changes
Undo
Open Project
Save Project
Attach Database
Close Database
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Table: Admin_tbl_product
Filter in any column
id
product_name
product_price
product_details
product_amzonlink
2 CeraVe Hydrating Cleanser 650 For Dry Skin Non-Foaming Face Wash … https://rb.gy/gq1bc7
4 Minimalist 5% Aquaporin Booster 300 Face Wash For Dry Skin with … https://rb.gy/40afke
5 Plum Green Tea 250 Pore Cleansing Face Wash for Dry … https://tinyurl.com/2krtr563
6 Dot & Key Skin Barrier Repair 500 Daily Skin Care Routine for Dry| Non… https://tinyurl.com/uvn39ecd
7 Deconstruct Hyaluronic Acid Hydratin… 300 Deconstruct Hyaluronic Acid Hydratin… https://www.amazon.in/Deconstruct-…
8 CeraVe Moisturizing Cream 1400 For Dry To Very Dry Skin - Formulate… https://www.amazon.in/CeraVe-…
9 Cetaphil Moisturising Lotion 1600 Moisturising Lotion for Face , dry … https://www.amazon.in/Cetaphil-…
10 Minimalist Marula Oil 5% 300 Face Moisturizer For Dry Skin With … https://www.amazon.in/Minimalist-…
11 The Derma Co 5% Nia-Ceramide Deep … 400 5% Nia-Ceramide Deep Moisturizing … https://www.amazon.in/Derma-Nia-…
12 Plum 2% Niacinamide Moisturizer Crea… 450 Super Light Gel for Dry Skin | Oil … https://www.amazon.in/Plum-…
13 Dot & Key Ceramides Moisturizer 350 Moisturizer with Hyaluronic for … https://www.amazon.in/Ceramides-…
14 Deconstruct Daily Hydrating Duo 450 Hydrating Face Wash + Hydrating Face… https://www.amazon.in/Daily-…
15 Minimalist Cream Sunscreen 400 Sunscreen SPF 50 Lightweight With … https://www.amazon.in/Minimalist-…
16 The Derma Co 1% Hyaluronic Sunscreen 550 Sunscreen Aqua Gel SPF 50 PA++++ I … https://www.amazon.in/Derma-Co-…
17 Plum 2% Niacinamide & Rice Water … 350 Sunscreen SPF 50 PA+++ | for Dry … https://www.amazon.in/Plum-…
18 Dot & Key Blueberry Hydrate Barrier … 400 Sunscreen Spf 50+,Pa++++|For Dry Ski… https://www.amazon.in/DOT-KEY-…
19 Deconstruct Brightening Sunscreen 400 Sunscreen SPF 50+ for Dry Skin | 8 … https://www.amazon.in/Deconstruct-…
20 CeraVe Foaming Cleanser 500 Cleanser For Oily Skin - … https://www.amazon.in/CeraVe-Foaming…
21 Cetaphil Oily Skin Cleanser 550 Daily Face Wash for Oily, Acne prone… https://www.amazon.in/Cetaphil-…
22 The Derma Co. 2% Salicylic Acid Gel … 300 Face Wash | For Oily Skin | With … https://www.amazon.in/Derma-…
23 Plum Simply Bright 2% Niacinamide … 300 Face Wash with Rice Water | … https://www.amazon.in/Plum-…
24 Deconstruct Salicylic Acid Oil … 300 Face Wash | 0.5% Salicylic Acid + 1%… https://www.amazon.in/Deconstruct-…
25 Bioderma Sebium Gel 800 Bioderma Sebium Gel Moussant … https://www.amazon.in/Bioderma-…
1 - 23 of 58
Go to: 1
Edit Database Cell
Mode: Text
Editing row=1, column=0
Type: Text / Numeric; Size: 1 character(s)
Apply
Remote
Identity
Select an identity to connect
Upload
DBHub.io
Local
Current Database
Name
Last modified
SQL Log
Plot
DB Schema
Remote
UTF-8

Figure 6.2: Product

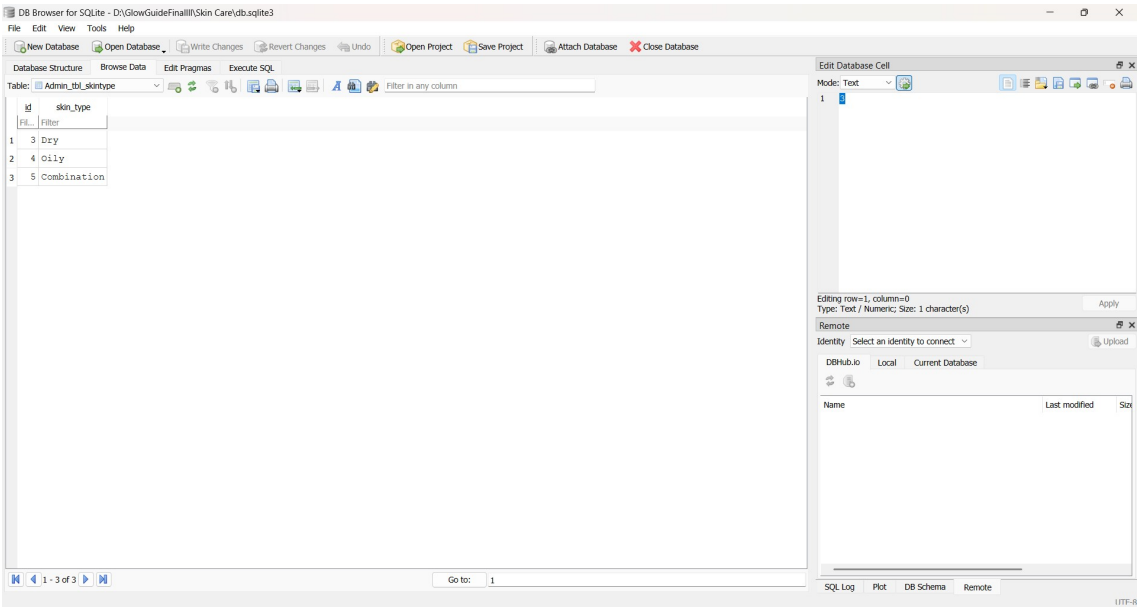
DB Browser for SQLite - D:\GlowGuideFinalIII\Skin Care\db.sqlite3
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database
Database Structure Browse Data Edit Pragmas Execute SQL
Table: Admin_tbl_skintype
Filter in any column
id skin_type
1 3 Dry
2 4 Oily
3 5 Combination
1 - 3 of 3
Go to: 1
Edit Database Cell
Mode: Text
Editing row=1, column=0
Type: Text / Numeric; Size: 1 character(s)
Apply
Remote
Identity Select an identity to connect
Upload
DBHub.io Local Current Database
Name Last modified
SQL Log Plot DB Schema Remote

Figure 6.3: Skin Type

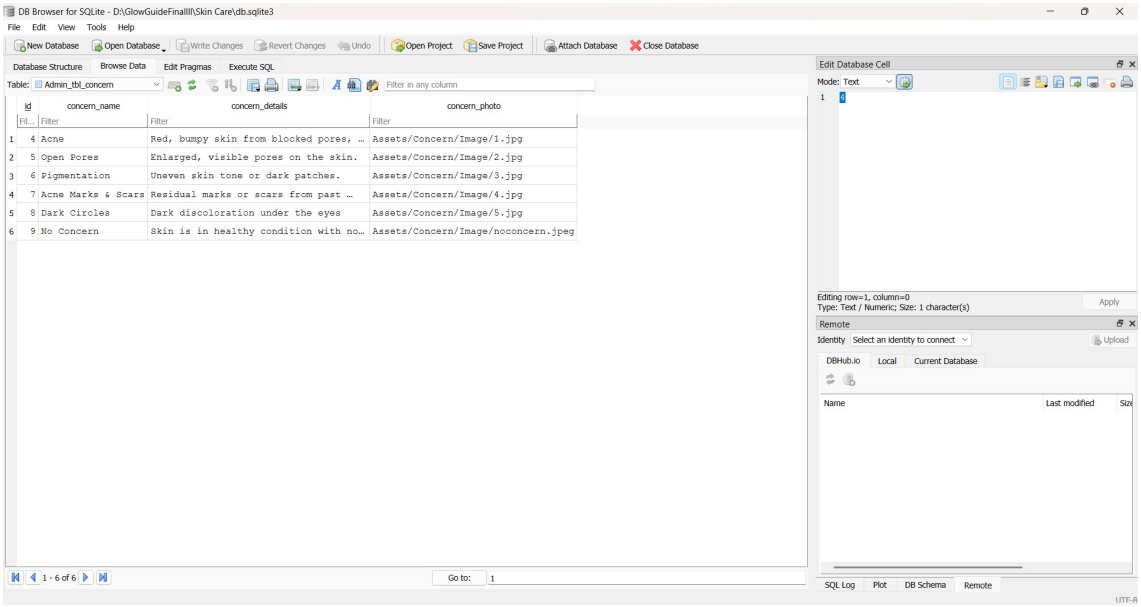
DB Browser for SQLite - D:\GlowGuideFinalIII\Skin Care\db.sqlite3
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database
Database Structure Browse Data Edit Pragmas Execute SQL
Table: Admin_tbl_concern
Filter in any column
id concern_name concern_details concern_photo
1 4 Acne Red, bumpy skin from blocked pores, … Assets/Concern/Image/1.jpg
2 5 Open Pores Enlarged, visible pores on the skin. Assets/Concern/Image/2.jpg
3 6 Pigmentation Uneven skin tone or dark patches. Assets/Concern/Image/3.jpg
4 7 Acne Marks & Scars Residual marks or scars from past … Assets/Concern/Image/4.jpg
5 8 Dark Circles Dark discoloration under the eyes Assets/Concern/Image/5.jpg
6 9 No Concern Skin is in healthy condition with no… Assets/Concern/Image/noconcern.jpeg
1 - 6 of 6
Go to: 1
Edit Database Cell
Mode: Text
Editing row=1, column=0
Type: Text / Numeric; Size: 1 character(s)
Apply
Remote
Identity Select an identity to connect
Upload
DBHub.io Local Current Database
Name Last modified
SQL Log Plot DB Schema Remote

Figure 6.4: Skin Concern

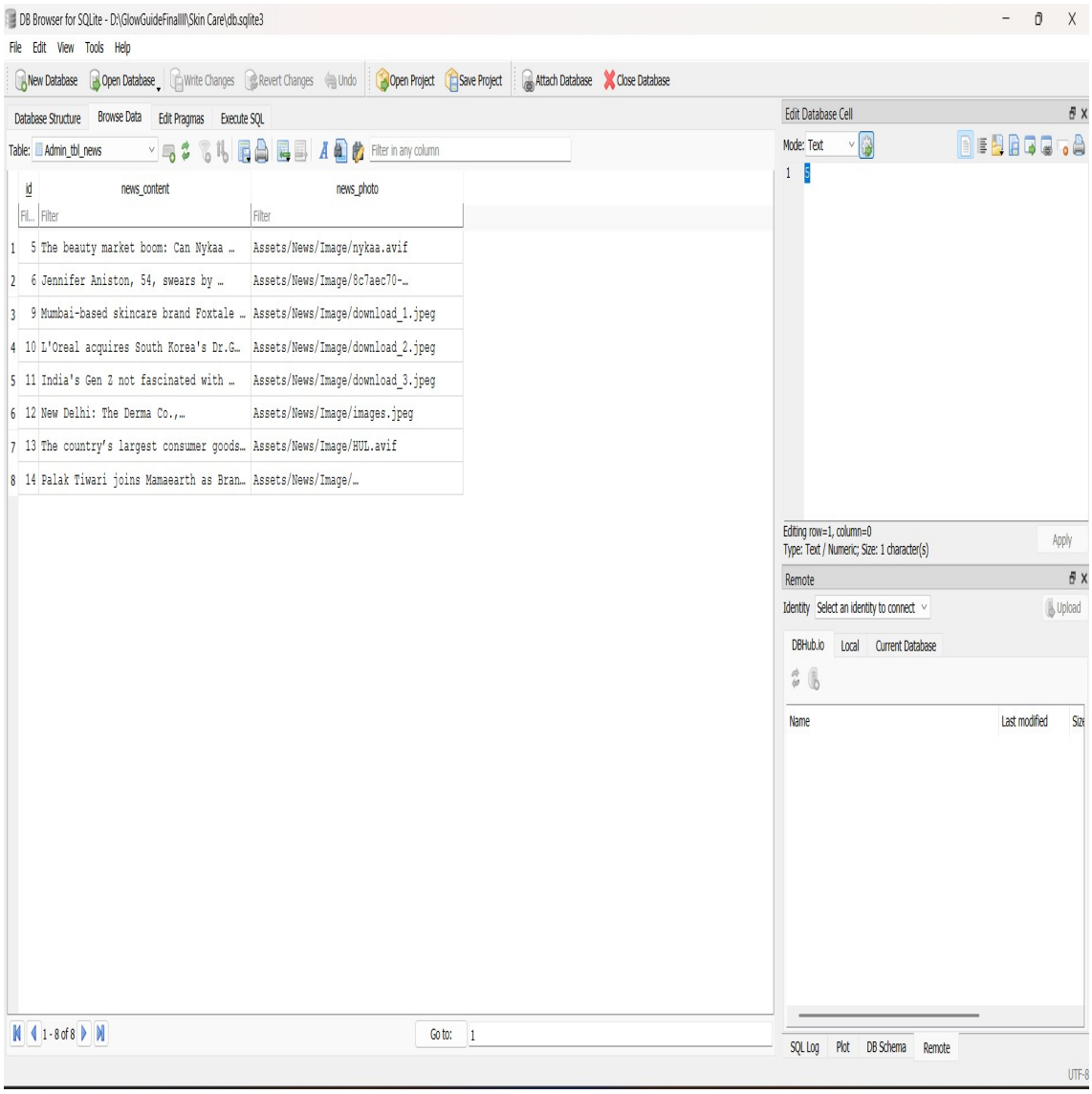

Figure 6.5: News

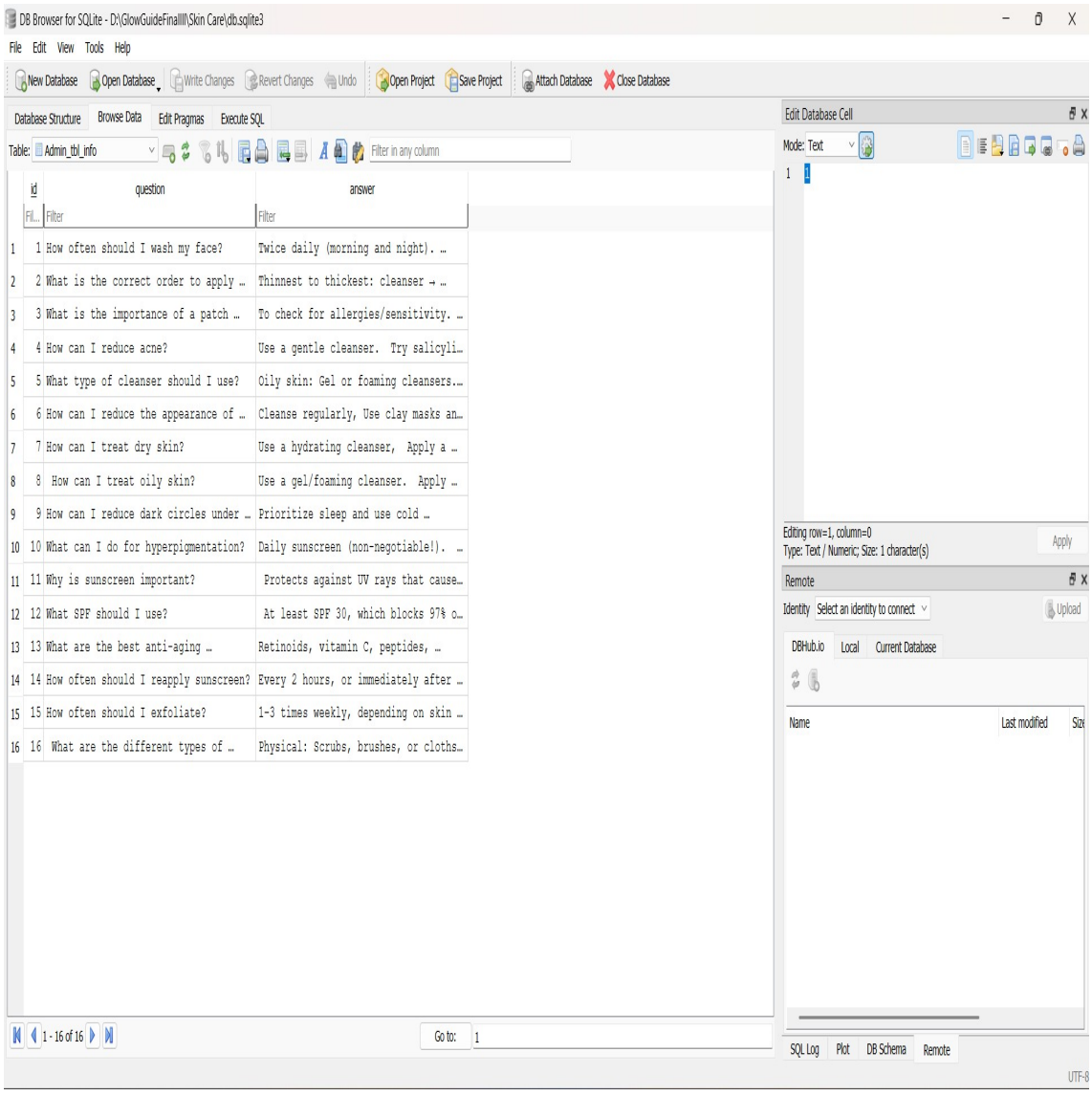
DB Browser for SQLite - D:\GlowGuideFinalllll\Skin Care\db.sqlite3
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database
Database Structure Browse Data Edit Pragmas Execute SQL
Table: Admin_tbl_info
Filter in any column
id question answer
1 How often should I wash my face? Twice daily (morning and night). …
2 What is the correct order to apply … Thinnest to thickest: cleanser → …
3 What is the importance of a patch … To check for allergies/sensitivity. …
4 How can I reduce acne? Use a gentle cleanser. Try salicyli…
5 What type of cleanser should I use? Oily skin: Gel or foaming cleansers.…
6 How can I reduce the appearance of … Cleanse regularly, Use clay masks an…
7 How can I treat dry skin? Use a hydrating cleanser, Apply a …
8 How can I treat oily skin? Use a gel/foaming cleanser. Apply …
9 How can I reduce dark circles under … Prioritize sleep and use cold …
10 What can I do for hyperpigmentation? Daily sunscreen (non-negotiable!). …
11 Why is sunscreen important? Protects against UV rays that cause…
12 What SPF should I use? At least SPF 30, which blocks 97% o…
13 What are the best anti-aging … Retinoids, vitamin C, peptides, …
14 How often should I reapply sunscreen? Every 2 hours, or immediately after …
15 How often should I exfoliate? 1-3 times weekly, depending on skin …
16 What are the different types of … Physical: Scrubs, brushes, or cloths…
1 - 16 of 16
Go to: 1
Edit Database Cell
Mode: Text
Editing row=1, column=0
Type: Text / Numeric; Size: 1 character(s)
Apply
Remote
Identity Select an identity to connect
Upload
DBHub.io Local Current Database
Name Last modified
SQL Log Plot DB Schema Remote
UTF-8

Figure 6.6: Question And Answer

6.2 Testing

6.2.1 Test Cases for Algorithm to Generate Personalized Skincare

1. Low Risk Alert
    - Input: Skin type ”Dry,” concern ”Acne” (matches available products).
    - Expected Output: No alert, suitable products displayed.
2. Moderate Risk Alert:
    - Input: Skin type ”Oily,” concern ”Hyperpigmentation” (product may cause minor irritation).
    - Expected Output: Moderate risk alert, suggests patch test.
3. High Risk Alert:
    - Input: Skin type ”Combination,” concern ”Acne” (product has harsh ingredients).
    - Expected Output: High risk alert, advises avoiding product.
4. Concern Not Added by Admin:
    - Input: User attempts to select concern ”Warts” (not added by admin; only admin-added concerns like ”Acne,” ”pigmentation” are available).
    - Expected Output: System restricts selection to admin-added concerns via a dropdown, preventing invalid input.

6.2.2 Test Cases for Algorithm to Manage Products from Admin Side:

1. Admin Adds Product:
    - Input: Add Vitamin C Serum for Hyperpigmentation.
    - Expected Output: Product added, confirmation displayed.
2. Admin Deletes Product
    - Input: Delete SPF 50+ Sunscreen.
    - Expected Output: Product removed, confirmation displayed..

6.2.3 Test Cases for Algorithm to View Recommendations on User Side:

1. User Views Recommendations:
    - Input: User requests recommendations
    - Expected Output: Products (e.g., face wash, moisturizer) with links displayed.

Chapter 7

CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

In conclusion, our Personalized Skincare Recommendation System is designed to make it easier for users to find skincare products that match their unique skin types and concerns. By creating a simple and user-friendly platform, we guide users through selecting their skin type, identifying specific issues, and receiving tailored product recommendations for essentials like face wash, moisturizer, sunscreen, and targeted treatments. With secure login and data protection, users can confidently explore personalized suggestions that address their skincare needs. Additionally, the system includes a complaint feature that allows users to report issues or dissatisfaction, enabling the admin to make necessary adjustments and improve the recommendations. Ultimately, our platform empowers users to make informed skincare choices, helping them achieve healthier skin and boosting their confidence.

7.2 Future Scope

The future of our Personalized Skincare Recommendation System looks promising, with plenty of opportunities to enhance the user experience and add valuable features. One exciting possibility is expanding the product database to cover a wider range of skincare concerns, ensuring that users receive even more accurate and personalized recommendations. We also plan to introduce a complaint system where users can report any issues with recommendations, allowing the admin to fine-tune the system and improve results. Another potential upgrade is integrating third-party APIs to provide real-time information on product availability, reviews, and pricing, helping users make well-informed choices. Additionally, we envision adding a personalized dashboard where users can track their skincare history and preferences over time. With these improvements, the system will continue to evolve, offering a seamless and personalized skincare journey for every user.

References

[1] Lee, G., Jiang, X. and Parde, N., 2023, December. A content-based skincare product recommendation system. In 2023 International Conference on Machine Learning and Applications (ICMLA) (pp. 2039-2043). IEEE.

[2] Lee, J., Yoon, H., Kim, S., Lee, C., Lee, J. and Yoo, S., 2024. Deep learning-based skin care product recommendation: A focus on cosmetic ingredient analysis and facial skin conditions. Journal of Cosmetic Dermatology, 23(6), pp.2066-2077.

[3] Lokesh, B., Devarakonda, A., Srinivas, G. and Naik, N.K., INTELLIGENT FACIAL SKIN CARE RECOMMENDATION SYSTEM.

[4] Tanveer, S., Khatoon, S., Begum, H.U. and Zainab, U., 2024. KOREAN SKIN CARE RECOMMENDATION SYSTEM. International Journal of Information Technology and Computer Engineering, 12(2), pp.797-807.

[5] Goh, C.L., Wu, Y., Welsh, B., Abad-Casintahan, M.F., Tseng, C.J., Sharad, J., Jung, S., Rojanamatin, J., Sitohang, I.B.S. and Chan, H.N.K., 2023. Expert consensus on holistic skin care routine: Focus on acne, rosacea, atopic dermatitis, and sensitive skin syndrome. Journal of Cosmetic Dermatology, 22(1), pp.45-54.

[6] Hemantha, L.M.I.T. & Gayathri, T.M.E. & Silva, N.N.M.. (2022). Personalized Smart Skincare Product Recommendation System. International Journal of Computer Applications. 184. 1-6. 10.5120/ijca2022922469.

[7] Lin, T.Y., Chan, H.T., Hsia, C.H. and Lai, C.F., 2022. Facial skincare products' recommendation with computer vision technologies. Electronics, 11(1), p.143.

[8] https://www.researchgate.net/directory/publications

[9] https://www.aad.org/public/everyday-care/skin-care-basics

[10] https://www.paulaschoice.com/ingredient-dictionary

[11] https://pubmed.ncbi.nlm.nih.gov/?term=skincare

[12] https://www.ewg.org/skindeep/

[13] https://www.cerave.com/

[14] https://chat.openai.com/

[15] https://www.statista.com/markets/415/topic/540/skin-care/

[16] https://www.wikipedia.org/

[17] https://draw.io/

[18] https://grok.com/

[19] https://chat.openai.com/

[20] https://www.python.org/doc/

[21] https://docs.djangoproject.com/en/5.1/

Appendix A

Implementation code

A.1 Admin Side

```
from django.shortcuts import render,redirect
from Admin.models import *
from User.models import *

def logout(request):
    del request.session['adminid']
    return redirect("Guest:Login")

def HomePage(request):
    if "adminid" in request.session:
        return render(request,"Admin/HomePage.html")
    else:
        return redirect("Guest:Login")

def Admin(request):
    # if "adminid" in request.session:
    if request.method=='POST':
        name=request.POST.get('name')
        email=request.POST.get('email')
        password=request.POST.get('password')
        tbl_admin.objects.create(admin_name=name,admin_email=email,
        admin_password=password)
        return render(request,"Admin/Admin.html")
```

```
    else:
        return render(request,"Admin/Admin.html")
    # else:
    #     return redirect("Guest:Login")

def Agelimit(request):
    if "adminid" in request.session:
        agedata=tbl_agelimit.objects.all()
        print(agedata)
        if request.method=='POST':
            agelimit=request.POST.get('limit')
            tbl_agelimit.objects.create(age_limit=agelimit)
            return redirect('Admin:Agelimit')
        else:
            return render(request,"Admin/Agelimit.html",{'age':agedata})
    else:
        return redirect("Guest:Login")

def editagelimit(request,eid):
    editdata=tbl_agelimit.objects.get(id=eid)
    if request.method =="POST":
        editdata.age_limit=request.POST.get('limit')
        editdata.save()
        return redirect('Admin:Agelimit')
    else:
        return render(request,"Admin/Agelimit.html",{'type':editdata})


def deleteagelimit(request,did):
    tbl_agelimit.objects.get(id=did).delete()
    return redirect('Admin:Agelimit')


def Category(request):
    if "adminid" in request.session:
        catdata=tbl_category.objects.all()
```

```
        print(catdata)
        if request.method=='POST':
            categoryname=request.POST.get('name')
            tbl_category.objects.create(category_name=categoryname)
            return redirect('Admin:Category')
        else:
            return render(request,"Admin/Category.html",{'cat':catdata})
    else:
        return redirect("Guest:Login")

def deletecategory(request,did):
    tbl_category.objects.get(id=did).delete()
    return redirect('Admin:Category')

def editcategory(request,eid):
    editdata=tbl_category.objects.get(id=eid)
    if request.method =="POST":
        editdata.category_name=request.POST.get('name')
        editdata.save()
        return redirect('Admin:Category')
    else:
        return render(request,"Admin/Category.html",{'type':editdata})



def Concern(request):
    if "adminid" in request.session:
        condata=tbl_concern.objects.all()
        print(condata)
        if request.method=='POST':
            concerns=request.POST.get('concern')
            photo=request.FILES.get('selconvern')
            details=request.POST.get('txt_details')
            tbl_concern.objects.create(concern_name=concerns,concern_photo=
            photo,concern_details=details)
            return redirect('Admin:Concern')
        else:
```

```
            return render(request,"Admin/Concern.html",{'con':condata})
    else:
        return redirect("Guest:Login")

def deleteconcern(request,did):
    tbl_concern.objects.get(id=did).delete()
    return redirect('Admin:Concern')

def editconcern(request,eid):
    editdata=tbl_concern.objects.get(id=eid)
    if request.method =="POST":
        editdata.concern_name=request.POST.get('concern')
        editdata.save()
        return redirect('Admin:Concern')
    else:
        return render(request,"Admin/Concern.html",{'type':editdata})

def District(request):
    if "adminid" in request.session:
        disdata=tbl_district.objects.all()
        print(disdata)
        if request.method=='POST':
            districtname=request.POST.get('name')
            tbl_district.objects.create(district_name=districtname)
            return redirect('Admin:District')
        else:
            return render(request,"Admin/District.html",{'dis':disdata})
    else:
        return redirect("Guest:Login")

def deletedistrict(request,did):
    tbl_district.objects.get(id=did).delete()
    return redirect('Admin:District')

def editdistrict(request,eid):
    editdata=tbl_district.objects.get(id=eid)
```

```
    if request.method =="POST":
        editdata.district_name=request.POST.get('name')
        editdata.save()
        return redirect('Admin:District')
    else:
        return render(request,"Admin/District.html",{'type':editdata})



def Gender(request):
    if "adminid" in request.session:
        gendata=tbl_gender.objects.all()
        print(gendata)
        if request.method=='POST':
            gendername=request.POST.get('gender')
            tbl_gender.objects.create(gender=gendername)
            return redirect('Admin:Gender')
        else:
            return render(request,"Admin/Gender.html",{'gen':gendata})
    else:
        return redirect("Guest:Login")

def deletegender(request,did):
    tbl_gender.objects.get(id=did).delete()
    return redirect('Admin:Gender')

def editgender(request,eid):
    editdata=tbl_gender.objects.get(id=eid)
    if request.method =="POST":
        editdata.gender=request.POST.get('gender')
        editdata.save()
        return redirect('Admin:Gender')
    else:
        return render(request,"Admin/Gender.html",{'type':editdata})

def Product(request):
    if "adminid" in request.session:
```

```
        concerns=tbl_concern.objects.all()
        productdata=tbl_product.objects.all()
        if request.method=='POST':
            name=request.POST.get('name')
            details=request.POST.get('details')
            price=request.POST.get('price')
            amlink=request.POST.get('amazonLink')
            fclink=request.POST.get('flipkartLink')
            concern=request.POST.get('sel_concern')
            tbl_product.objects.create(product_name=name,
                                       product_price=price,
                                       product_details=details,
                                       product_amzonlink=amlink,
                                       product_flpctlink=fclink,
                                       concern=tbl_concern.objects.get(id=
                                       concern))
            return redirect('Admin:Product')
        else:
            return render(request,"Admin/Product.html",
            {'pro':productdata,'concern':concerns})
    else:
        return redirect("Guest:Login")

def deleteproduct(request,did):
    tbl_product.objects.get(id=did).delete()
    return redirect('Admin:Product')

def editproduct(request,eid):
    editdata=tbl_product.objects.get(id=eid)
    if request.method =="POST":
        editdata.product_name=request.POST.get('name')
        editdata.product_details=request.POST.get('details')
        editdata.product_price=request.POST.get('price')
        editdata.product_amzonlink=request.POST.get('amazonLink')
        editdata.product_flpctlink=request.POST.get('flipkartLink')
        editdata.save()
```

```
        return redirect('Admin:Product')
    else:
        return render(request,"Admin/Product.html",{'type':editdata})

def gallery(request,id):
    imgdata=tbl_gallery.objects.filter(product=id)
    if request.method=='POST':
        img=request.FILES.get('photo')
        productid=tbl_product.objects.get(id=id)
        tbl_gallery.objects.create(gallery_image=img,product=productid)
        return redirect('Admin:gallery',id)
    else:
        return render(request,"Admin/Gallery.html",{'img':imgdata})

def delproductphoto(request,did):
    id=tbl_gallery.objects.get(id=did).product.id
    tbl_gallery.objects.get(id=did).delete()
    return redirect('Admin:gallery',id)

def SkinProduct(request,id):
    if "adminid" in request.session:
        skinproduct=tbl_skintype.objects.all()
        sknprodata=tbl_skinproduct.objects.filter(product=id)
        if request.method=='POST':
            skinname=request.POST.get('skinType')
            tbl_skinproduct.objects.create(skinproduct_type=tbl_skintype.obje
            cts.get(id=skinname),product=tbl_product.objects.get(id=id))
            return redirect('Admin:SkinProduct',id)
        else:
            return render(request,"Admin/SkinProduct.html",
            {'sknpro':sknprodata,'skintype':skinproduct,'pid':id})
    else:
        return redirect("Guest:Login")

def deleteskinproduct(request,did,pid):
    tbl_skinproduct.objects.get(id=did).delete()
```

```
    return redirect('Admin:SkinProduct',pid)

def editskinproduct(request,eid,pid):
    skintype=tbl_skintype.objects.all()
    editdata=tbl_skinproduct.objects.get(id=eid)
    if request.method =="POST":
        editdata.skinproduct_type=tbl_skintype.objects.get(id=request.POST.
        get("skin_type"))
        editdata.save()
        return redirect('Admin:SkinProduct',pid)
    else:
        return render(request,"Admin/SkinProduct.html",
        {'edit':editdata,'skintype':skintype,'pid':pid})

def Info(request):
    if "adminid" in request.session:
        return render(request,"Admin/Info.html")
    else:
        return redirect("Guest:Login")

def Skintype(request):
    if "adminid" in request.session:
        skndata=tbl_skintype.objects.all()
        print(skndata)
        if request.method=='POST':
            skintype=request.POST.get('skintype')
            tbl_skintype.objects.create(skin_type=skintype)
            return redirect('Admin:Skintype')
        else:
            return render(request,"Admin/Skintype.html",{'skn':skndata})
    else:
        return redirect("Guest:Login")

def deleteskintype(request,did):
    tbl_skintype.objects.get(id=did).delete()
    return redirect('Admin:Skintype')
```

```
def editskintype(request,eid):
    editdata=tbl_skintype.objects.get(id=eid)
    if request.method == "POST":
        editdata.skin_type=request.POST.get('skintype')
        editdata.save()
        return redirect('Admin:Skintype')
    else:
         return render(request,"Admin/Skintype.html",{'type':editdata})

def News(request):
    if "adminid" in request.session:
        newsdata=tbl_news.objects.all()
        print(newsdata)
        if request.method=='POST':
            newscontent=request.POST.get('content')
            newsphoto=request.FILES.get('photo')
            tbl_news.objects.create(news_content=newscontent,news_photo=newsphoto)
            return redirect('Admin:News')
        else:
            return render(request,"Admin/News.html",{'new':newsdata})
    else:
        return redirect("Guest:Login")

def deletenews(request,did):
    tbl_news.objects.get(id=did).delete()
    return redirect('Admin:News')

def ProductSkintype(request,id):
    if "adminid" in request.session:

        skintype=tbl_skintype.objects.all()
        proskndata=tbl_productskintype.objects.filter(product=id)
        print(proskndata)
        if request.method=='POST':
            skinname=request.POST.get('skin_type')
```

```
            tbl_productskintype.objects.create(product_skintype=tbl_skintype.
            objects.get(id=skinname),product=tbl_product.objects.get(id=id))
            return redirect('Admin:ProductSkintype',id)
        else:
            return render(request,"Admin/ProductSkintype.html",
            {'proskn':proskndata,'skintype':skintype,'pid':id})
    else:
        return redirect("Guest:Login")

def deleteproductskintype(request,did,pid):
    tbl_productskintype.objects.get(id=did).delete()
    return redirect('Admin:ProductSkintype',pid)

def editproductskintype(request,eid,pid):
    skintype=tbl_skintype.objects.all()
    editdata=tbl_productskintype.objects.get(id=eid)
    if request.method =="POST":
        editdata.product_skintype=tbl_skintype.objects.get(id=request.POST.
        get("skin_type"))
        editdata.save()
        return redirect('Admin:ProductSkintype',pid)
    else:
        return render(request,"Admin/ProductSkintype.html",
        {'edit':editdata,'skintype':skintype,'pid':pid})

def Info(request):
    if "adminid" in request.session:
        infodata=tbl_info.objects.all()
        print(infodata)
        if request.method=='POST':
            inquestion=request.POST.get('question')
            inanswer=request.POST.get('answer')
            tbl_info.objects.create(question=inquestion,answer=inanswer)
            return redirect('Admin:Info')
        else:
            return render(request,"Admin/Info.html",{'inf':infodata})
```

```
    else:
        return redirect("Guest:Login")

def deleteinfo(request,did):
    tbl_info.objects.get(id=did).delete()
    return redirect('Admin:Info')

def editinfo(request,eid):
    editdata=tbl_info.objects.get(id=eid)
    if request.method =="POST":
        editdata.question=request.POST.get('question')
        editdata.answer=request.POST.get('answer')
        editdata.save()
        return redirect('Admin:Info')
    else:
        return render(request,"Admin/Info.html",{'type':editdata})

def ViewComplaint(request):
    if "adminid" in request.session:
        complaintdata=tbl_complaint.objects.all()
        return render(request,"Admin/ViewComplaint.html,{'comp':complaintdata})
    else:
        return redirect("Guest:Login")

def Reply(request,id):
    replydata=tbl_complaint.objects.get(id=id)
    if request.method=='POST':
        reply=request.POST.get('content')
        replydata.reply=reply
        replydata.save()
        return redirect('Admin:ViewComplaint')
    else:
        return render(request,'Admin/Reply.html',{'reply':replydata})
```

A.2 User Side

```
from django.shortcuts import render,redirect
from Guest.models import *
from Admin.models import *
from User.models import *

# Create your views here.

def logout(request):
    del request.session['userid']
    return redirect("Guest:Login")

def HomePage(request):
    if "userid" in request.session:
        return render(request, 'User/HomePage.html')
    else:
        return redirect("Guest:Login")

def Userconcern(request):
    userconcerndata=tbl_userconcern.objects.all()
    if "userid" in request.session:
        sknconcern=tbl_concern.objects.all()
        userid=tbl_user.objects.get(id=request.session['userid'])
        print(sknconcern)
        if request.method=='POST':
            usrcncrn=request.POST.get('skin_type')
            tbl_userconcern.objects.create(concern=tbl_concern.objects.get
            (id=request.POST.get('concern')),user=userid)
            return redirect('User:Userconcern')
        else:
            return render(request,"User/Userconcern.html",
            {'skinconcern':sknconcern,'pid':id,'data':userconcerndata})
    else:
        return redirect("Guest:Login")
```

```
def deleteuserconcern(request,did):
    tbl_userconcern.objects.get(id=did).delete()
    return redirect('User:Userconcern')


def Complaint(request):
    if "userid" in request.session:
        return render(request, 'User/Complaint.html')
    else:
        return redirect("Guest:Login")

def MyProfile(request):
    if "userid" in request.session:
        userdata=tbl_user.objects.get(id=request.session['userid'])
        return render(request, 'User/MyProfile.html',{'data':userdata})
    else:
        return redirect("Guest:Login")

def EditProfile(request):
    if "userid" in request.session:
        userdata=tbl_user.objects.get(id=request.session['userid'])
        if request.method=='POST':
            userdata.user_name=request.POST.get('name')
            userdata.user_email=request.POST.get('email')
            userdata.user_contact=request.POST.get('contact')
            userdata.user_address=request.POST.get('address')
            userdata.save()
            return render(request, 'User/EditProfile.html',{'msg':'Profile
            Updated','data':userdata})
        else:
            return render(request, 'User/EditProfile.html',{'data':userdata})
    else:
        return redirect("Guest:Login")

def profile(request):
```

```
    if "userid" in request.session:
        userdata=tbl_user.objects.get(id=request.session['userid'])
        if request.method=='POST':
            userdata.user_photo=request.FILES.get('photo')
            userdata.save()
            return render(request, 'User/ChoosePhoto.html',{'msg':'Profile
            Updated','data':userdata})
        else:
            return render(request, 'User/ChoosePhoto.html',{'data':userdata})
    else:
        return redirect("Guest:Login")

def ChangePassword(request):
    if "userid" in request.session:
        userdata=tbl_user.objects.get(id=request.session['userid'])
        if request.method=='POST':
            dbpassword=userdata.user_password
            currentpassword=request.POST.get('txt_current')
            newpassword=request.POST.get('txt_new')
            confirmpassword=request.POST.get('txt_confirm')
            if currentpassword==dbpassword:
                if newpassword==confirmpassword:
                    userdata.user_password=newpassword
                    userdata.save()
                    return render(request, 'User/ChangePassword.html',
                    {'msg1':'Password Changed'})
                else:
                    return render(request, 'User/ChangePassword.html',
                    {'msg':'New Password and Confirm Password not matched'})
            else:
                return render(request, 'User/ChangePassword.html',
                {'msg':'Current Password not matched'})
        else:
            return render(request, 'User/ChangePassword.html')
    else:
        return redirect("Guest:Login")
```

```
def ProductSuggestion(request):
    userconcern=tbl_concern.objects.all()
    skintype=tbl_skintype.objects.all()
    if "userid" in request.session:
        suggdata=tbl_product.objects.all()
        image=tbl_gallery.objects.filter(product__in=suggdata)
        return render(request,"User/ProductSuggestion.html",'inf':suggdata,
        'img':image, 'skinconcern':userconcern, 'skintype':skintype})
    else:
        return redirect("Guest:Login")

def ViewInfo(request):
    if "userid" in request.session:
        infodata=tbl_info.objects.all()
        return render(request,"User/ViewInfo.html",{'inf':infodata})
    else:
        return redirect("Guest:Login")

def ViewNews(request):
    if "userid" in request.session:
        newsdata=tbl_news.objects.all()
        return render(request,"User/ViewNews.html",{'new':newsdata})
    else:
        return redirect("Guest:Login")

def ajaxsearchproduct(request):
    if request.GET.get("skinid") != "" and request.GET.get("concernid") !="":
        skinpdt = tbl_product.objects.filter(tbl_productskintype__
        product_skintype=request.GET.get("skinid"))
        con = tbl_concern.objects.get(id=request.GET.get("concernid"))
        if con.concern_name == "No Concern":
            concernpdt = []
        else:
            concernpdt = tbl_product.objects.filter(concern=request.GET.
            get("concernid"))
```

```
            return render(request,"User/AjaxSearchProduct.html",
            {'skindata':skinpdt,"concerndata":concernpdt})
    elif request.GET.get("skinid") != "":
        skinpdt = tbl_product.objects.filter(tbl_productskintype__product_
        skintype=request.GET.get("skinid"))
        return render(request,"User/AjaxSearchProduct.html",
        {'skindata':skinpdt})
    elif request.GET.get("concernid") != "":
        con = tbl_concern.objects.get(id=request.GET.get("concernid"))
        if con.concern_name == "No Concern":
            concernpdt = []
        else:
            concernpdt = tbl_product.objects.filter(concern=request.GET.
            get("concernid"))
            return render(request,"User/AjaxSearchProduct.html",
            {'concerndata':concernpdt})
    else:
        pdt = tbl_product.objects.all()
        return render(request,"User/AjaxSearchProduct.html",{'data':pdt})

def Complaint(request):
    if "userid" in request.session:
        userid=tbl_user.objects.get(id=request.session['userid'])
        complaintdata=tbl_complaint.objects.all()
        if request.method=='POST':
            comptitle=request.POST.get('title')
            compcontent=request.POST.get('content')
            tbl_complaint.objects.create(title=comptitle,content=compcontent,
            user=userid)
            return redirect('User:Complaint')
        else:
            return render(request,"User/Complaint.html",{'comp':complaintdata})
    else:
        return redirect("Guest:Login")

def routine(request):
```

```
    if "userid" in request.session:
        return render(request, 'User/routine.html')
    else:
        return redirect("Guest:Login")

def tips(request):
    if "userid" in request.session:
        return render(request, 'User/tips.html')
    else:
        return redirect("Guest:Login")

def Ingredient(request):
    if "userid" in request.session:
        return render(request, 'User/Ingredient.html')
    else:
        return redirect("Guest:Login")

def Findskin(request):
    if "userid" in request.session:
        return render(request, 'User/Findskin.html')
    else:
        return redirect("Guest:Login")
```

Appendix B

Screenshots

B.1 Admin Home Page

The admin dashboard of our Personalized Skincare Recommendation System. On the left, there's a sidebar menu with easy options like Home Page, Basic Data, Product Entry, News, Complaint, and FAQs, making it simple for admins to jump between tasks. Up top on the right, you'll see the admin's profile picture and name, confirming they're logged in securely. Built with Django and session-based login, this dashboard keeps things organized and safe, supporting the system's skincare recommendations.

Figure B.1: Admin Dashboard

B.2 Admin Product Entry

It shows the admin Product Entry page of our Personalized Skincare Recommendation System, for adding new products. It features a simple form where admins can input product details such as name, details, concern, price, Amazon link, and Flipkart link, with a Submit button to save the data.

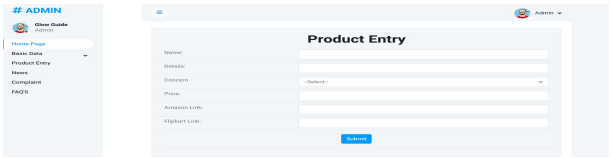

Figure B.2: Admin-Product Entry

B.3 Admin Skin Type Entry

It displays the admin skin type entry page of our Personalized Skincare Recommendation System. The main area features a form where admins can enter a new skin type and submit it with a click, alongside a table listing existing types—Dry, Oily, and Combination—with options to delete or edit each entry.

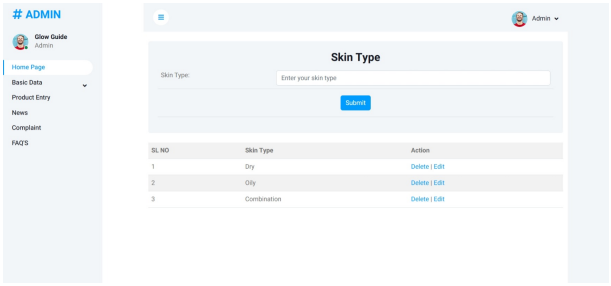

Figure B.3: Admin-Skin Type Entry

B.4 Admin Skin concern Entry

It highlights the Skin Concern page of our Personalized Skincare Recommendation System. The main section features a form where admins can add a new skin concern by typing the name, adding details, and uploading a photo, with a Submit button to save it. Below, a table lists existing concerns like Acne, Open Pores, Pigmentation, Acne Marks & Scars, Dark Circles, and No Concern, each with a photo and details, plus Delete and Edit options.

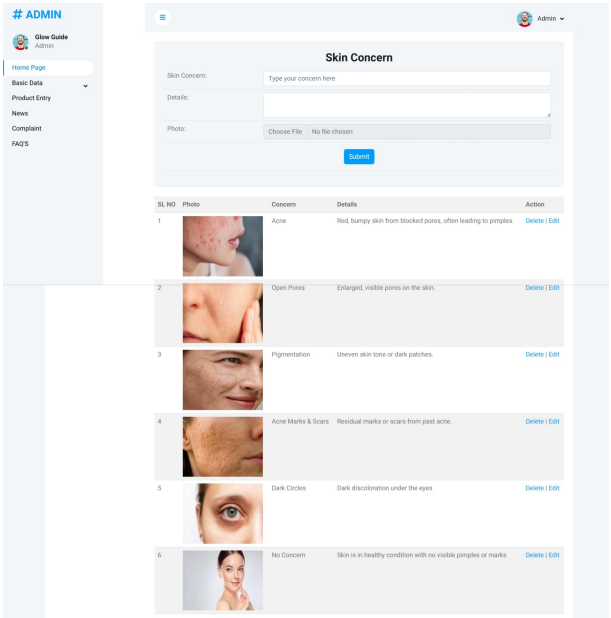

Figure B.4: Admin-Skin Concern Entry

B.5 Admin Complaint Page

This screenshot displays the admin dashboard where complaints submitted by users and can be viewed and managed by admin. The table shows columns for serial number, title, content, reply, and action, although no data is currently listed. It helps the admin monitor and respond to customer issues efficiently.

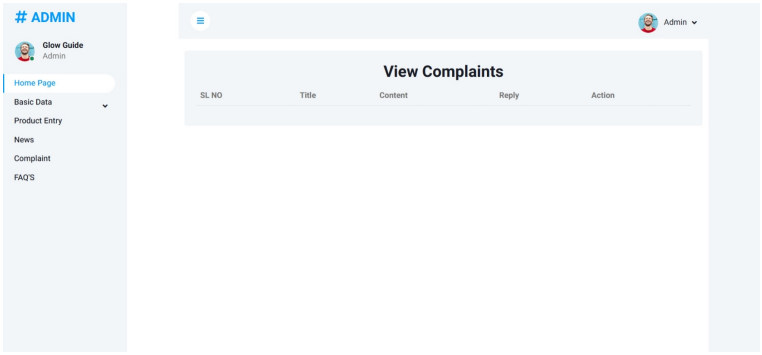

Figure B.5: Complaint

B.6 User Home Page

The user homepage of the application, titled Glow Guide, serves as the main interface for users. It showcases trending skincare products, features like an ingredient checker and routine builder, the latest news in the skincare industry, FAQs, and tips for skin care. The page emphasizes both usability and visual appeal, making the platform user-friendly and informative.

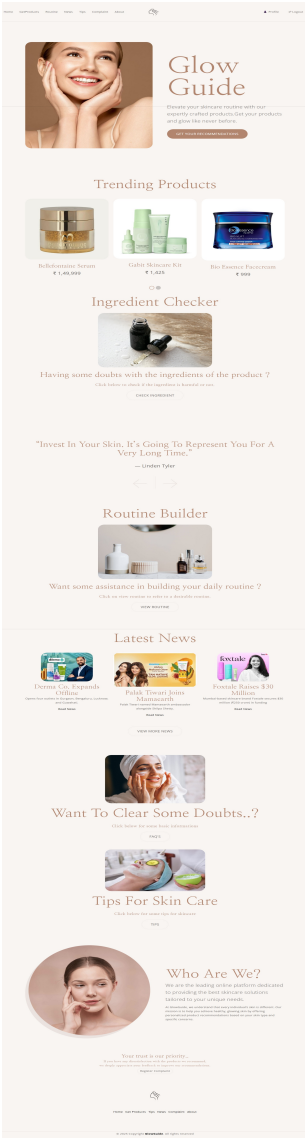
Glow Guide
Trending Products
Ingredient Checker
Having some doubts with the ingredients of the product ?
"Invest In Your Skin. It's Going To Represent You For A Very Long Time."
Routine Builder
Want some assistance in building your daily routine ?
Latest News
Want To Clear Some Doubts..?
Tips For Skin Care
Who Are We?

Figure B.6: User Home Page

B.7 User Sign Up Page

This page is designed for new users to register on the platform. It includes fields for name, email, contact, address, photo upload, gender, age, and password. Once filled, users can submit their details to create an account or cancel the registration process if needed.

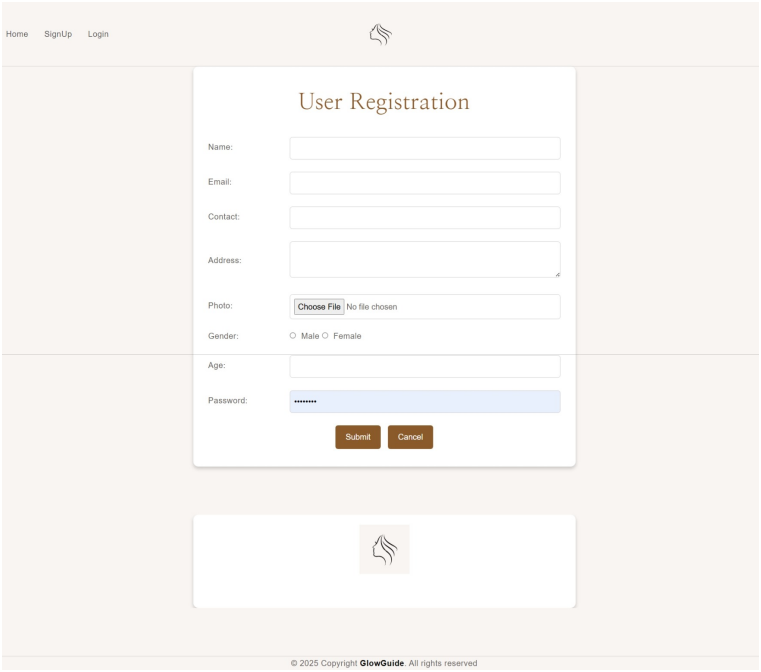

Figure B.7: Sign Up Page

B.8 User Login

The login page allows registered users to securely access their accounts by entering their email and password. It features a clean and minimalistic design with easy navigation options for home, signup, and login.

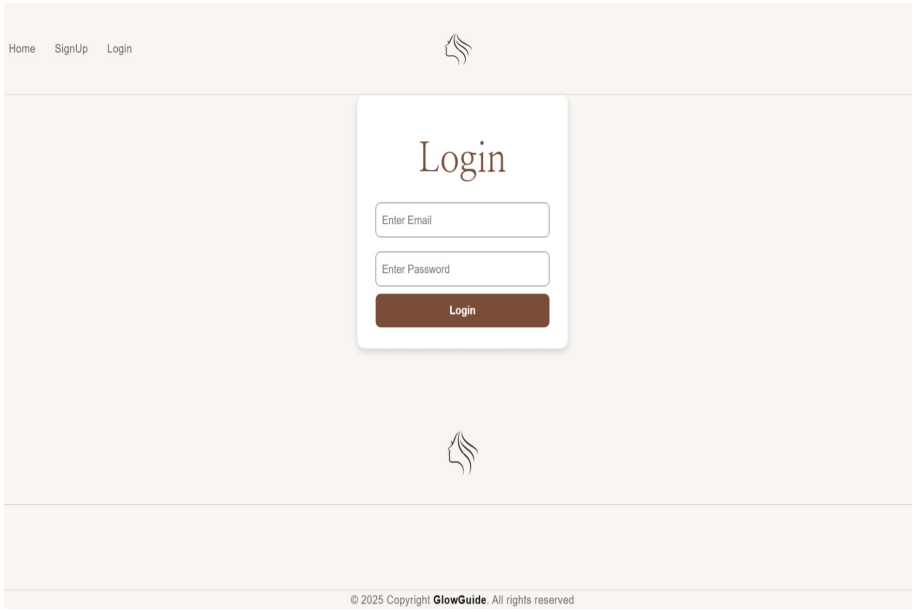

Figure B.8: Login Page

B.9 Skincare Product Recommendations

This section of the application provides personalized product recommendations based on the user's selected skin type and skin concern. Users first identify their skin type (dry, combination, or oily) with the help of a visual guide and clear instructions. Then, they choose a specific skin concern such as acne, pigmentation, or open pores. Based on these inputs, the system suggests suitable skincare products tailored to their individual needs, enhancing the relevance and effectiveness of the skincare routine.

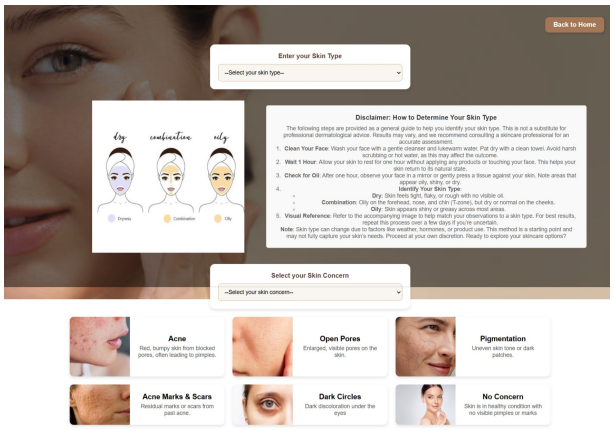

Figure B.9: Skincare Product Recommendations

B.10 Ingredient Checker

The Skincare Ingredient Checker enables users to verify whether a specific ingredient in their skincare products is potentially harmful. By entering an ingredient name (e.g., Parabens or Fragrance), users receive information about its safety. This tool promotes informed decision-making and encourages users to be more conscious of the ingredients they apply to their skin.

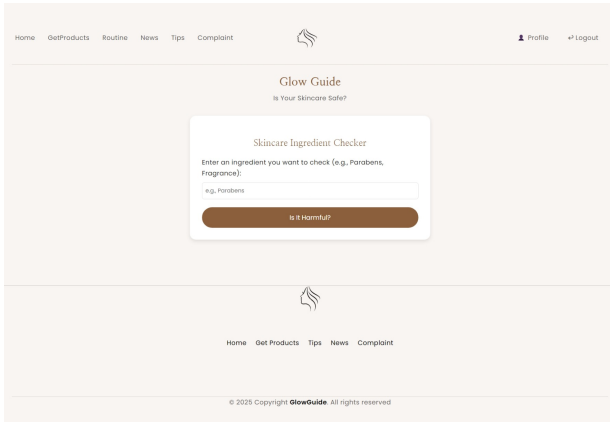

Figure B.10: Check Ingredient

B.11 User Complaint

This page provides users with a dedicated platform to share their feedback or report any issues encountered while using the application. The form includes fields for a complaint title and detailed content, allowing users to clearly express their concerns. Once submitted, the complaints can be reviewed and addressed by the system administrators, ensuring continuous improvement and enhanced user satisfaction.

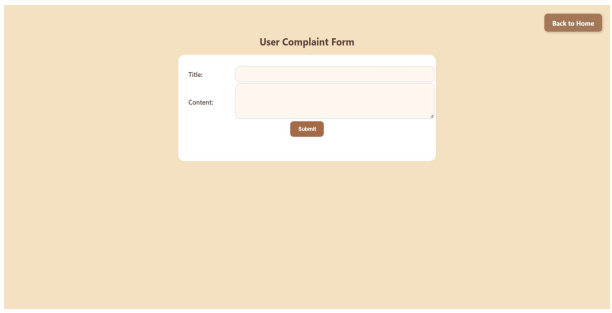

Figure B.11: User Complaints

B.12 Skincare News

The Latest News section provides users with up-to-date information from the beauty and skincare industry. It features articles on market trends, brand updates, product launches, and celebrity skincare routines. Each news item is accompanied by a relevant image and summary, helping users stay informed about the latest happenings and innovations in skincare and wellness.

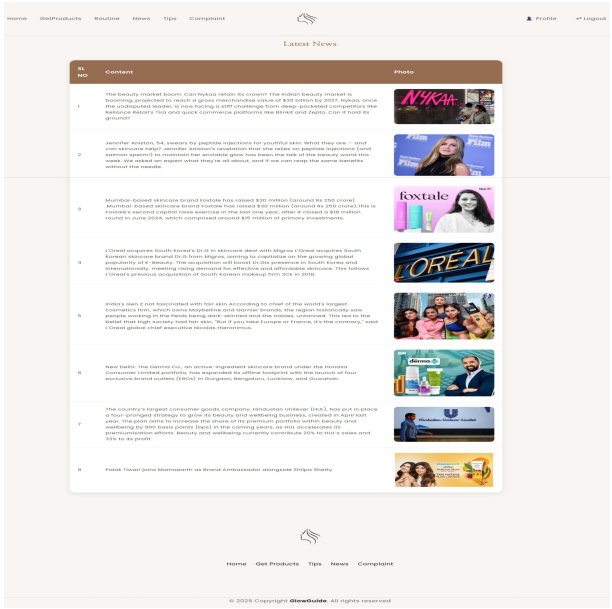

Figure B.12: Skincare news

B.13 Skincare Tips

This page offers users a comprehensive guide to skincare, featuring categorized tips to help maintain healthy and radiant skin. It includes general skincare practices along with additional categories such as exfoliation, diet and hydration, lifestyle, and targeted skin concerns. Each section is collapsible, allowing users to explore advice tailored to different aspects of skin health. This feature enhances user engagement and provides valuable educational content.

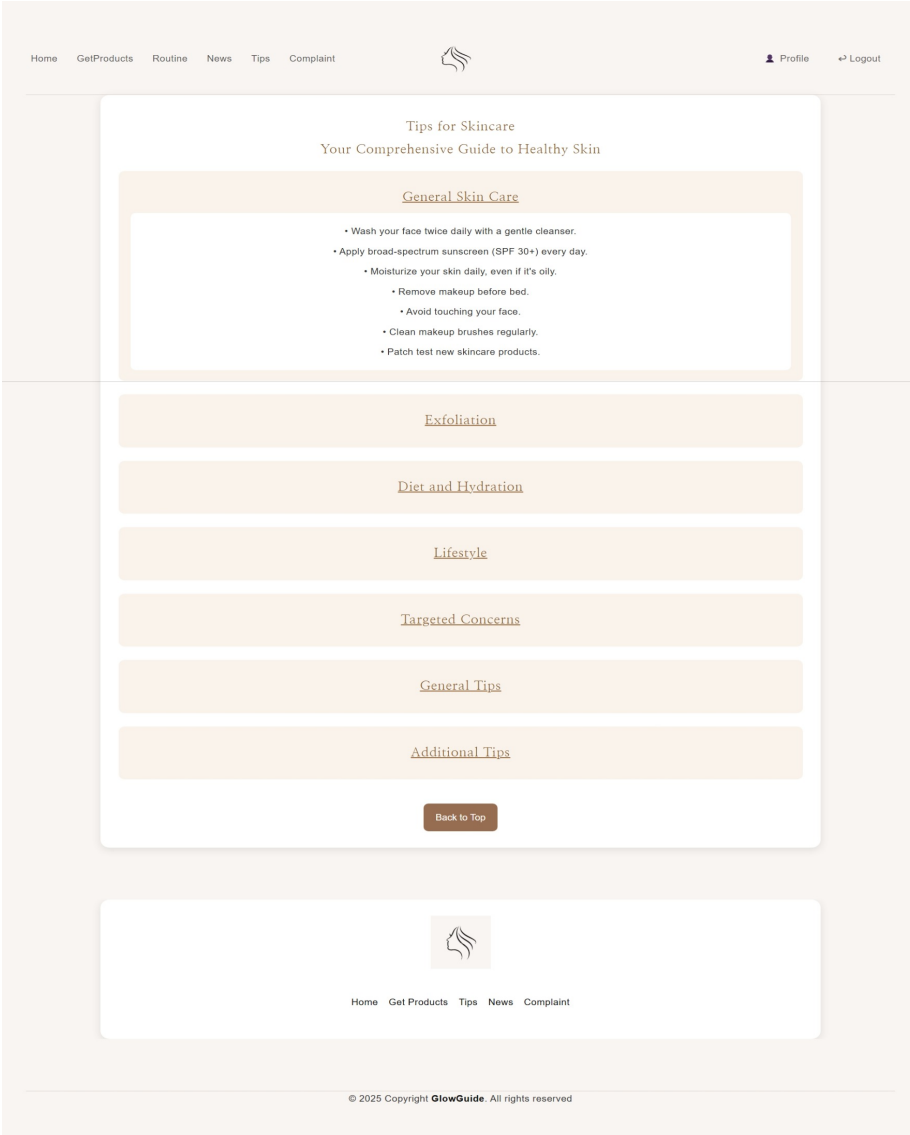

Figure B.13: Skincare Tips